

# **Vignette Application Portal**

## VAP 4.1 Sun/WebLogic Performance Benchmark



Vignette Corporation  
1601 South MoPac Expressway  
Austin, TX 78746-5776  
Phone: 1.512.741.4300  
Fax: 1.512.741.1403  
<http://www.vignette.com>

Copyright © 2003 Vignette Corporation. All rights reserved. Vignette and the V Logo are trademarks or registered trademarks of Vignette Corporation in the United States and other countries. All other company, product, and service names or brands are the trademarks or registered trademarks of their respective owners. U.S. Patent No. 6,327,628 and Patents Pending.

## Audience

The document is intended for a technical audience that is planning a VAP implementation. Vignette recommends consulting with Vignette Professional Services who can assist with the specific details of individual implementation architectures.

## Disclaimer

Vignette does not warrant, guarantee, or make representations concerning the contents of this document. All information is provided "AS-IS," without express or implied warranties of any kind. Vignette reserves the right to change the contents of this document at any time without obligation to notify anyone of such changes.

Vignette certifies multiple VAP system configurations. Vignette only certifies platforms that pass rigorous internal testing. Vignette strongly recommends that customers use VAP on certified platforms only. The following documentation provides performance benchmark testing for a single sample configuration. For a list of supported configurations, see the *Supported Platforms Matrix for Vignette Application Portal* on [VOLSS](#) (Vignette On-Line Support System).

Note that using a certified configuration does not guarantee that you will achieve the results documented herein. There may be parameters or variables that were not contemplated during these benchmarking and performance tests.

For any VAP production deployment, Vignette recommends a rigorous performance evaluation of the environment and application to ensure that there are no system, configuration or custom development bottlenecks present hindering the overall performance of the portal.

## Table of Contents

<b>Executive Summary .....</b>	<b>4</b>
<b>Introduction.....</b>	<b>5</b>
<b>Testing Methodology .....</b>	<b>6</b>
Performance Metrics & Terminology .....	8
<b>Test Architecture &amp; Topology .....</b>	<b>9</b>
<b>Test Scenarios .....</b>	<b>11</b>
<b>Performance Analysis .....</b>	<b>13</b>
<b>Baseline Test Results .....</b>	<b>17</b>
Baseline Guest Test .....	17
Baseline Login Test.....	18
<b>Benchmark Test Results.....</b>	<b>19</b>
Nominal Load Test.....	19
Increasing Load Test.....	20
Peak Usage Load Test .....	22
Scalability Test .....	23
Longevity Test .....	24
<b>A1 – Portal Configuration .....</b>	<b>26</b>
<b>A2 – Test Systems Specification .....</b>	<b>32</b>
Application Servers.....	32
Database Server & External Web Server .....	32
Load Agents & Controller.....	33
<b>A3 – System Tuning Guide.....</b>	<b>34</b>
<b>A4 – Performance Best Practices .....</b>	<b>39</b>
<b>A5 – Detailed Test Scenarios .....</b>	<b>43</b>
Scenarios.....	43
Procedures .....	45

---

## Executive Summary

---

This document presents a performance benchmark report on Vignette Application Portal (VAP) 4.1 SP1<sup>1</sup>. The primary objective of this document is to provide a performance analysis of VAP in a realistic enterprise production deployment using a number of realistic end user scenarios.

A number of tests have been carried out, which can be broadly categorized into baseline tests and benchmark tests. Baseline tests are used to ensure the production environment and VAP are configured and working as expected. The results of the baseline tests also provide an indication of the optimal performance attainable by the application server in isolation from VAP. Consequently, these results should be used in any comparative analysis of the actual benchmark test results.

The benchmark tests focus purely on the performance of the portal in a range of operating conditions. Performance tests have been carried out under nominal loads to measure performance at non-peak periods, increasing workloads to determine the maximum number of concurrent users VAP can support and peak period workloads to measure the performance of VAP during periods of peak usage. Additionally a series of scalability tests and a longevity test have been completed to determine the scalability of the VAP architecture and its stability and consistency in performance over a prolonged period of time.

For nominal load conditions the average response time for all scenarios was below 0.5 seconds, highlighting fast response times during periods of non-peak usage. The results of the increasing workload and peak period tests provide useful capacity planning information. For the overall scenario, combining both guest users and registered users, VAP has been shown to support approximately 143 concurrent users per CPU (750 MHz SPARC processor with 30 sec mean think time) with acceptable response times of 4 seconds or less at peak periods. In terms of server throughput this corresponds to approximately 4.3 PV/sec/CPU. This demonstrates good performance when compared to the corresponding throughput of 17.6 PV/sec/CPU for the baseline tests where the application server is serving a static JSP page.

In terms of scalability the benchmark results illustrate near linear scalability of VAP when tested against 1, 2 and 4 CPUs successively. Finally, the longevity test demonstrates the stability and consistent performance of VAP over a period of 8 hours at 70% of peak usage.

This document hopefully presents all the necessary information for parties interested in reproducing these test conditions. The test architecture is fully described along with detailed information regarding the VAP deployment, configuration and the actual end user scenarios being used as part of the overall benchmark tests.

---

<sup>1</sup> VAP 4.1 SP1 (build 26)

---

## Introduction

---

### Objectives

The objective of the performance benchmark is to measure the performance of Vignette Application Portal (VAP) 4.1 in a typical<sup>2</sup> production environment with realistic usage scenarios. Further, an analysis of the performance results is presented with guidelines for the configuration and tuning of the system to achieve these results.

The results of the performance benchmark provide key benchmark metrics for Vignette customers to assist in their capacity planning process and in determining a suitable production architecture to support actual performance requirements. The report will also be of use for prospective customers in evaluating the performance of Vignette Application Portal (VAP) 4.1.

When analyzing the results of the benchmark tests the reader should examine the results of the baseline tests for comparative study.

---

### Document Overview

The document is divided into a number of sections. A brief summary of each is provided below:

- [Testing Methodology](#): Introduces the testing methodology, discusses some of the terminology associated with the performance testing and the metrics used to analyze the performance of the portal.
- [Test Architecture](#): Presents a high level overview of the test environment and setup used during the benchmark testing.
- [Test Scenarios](#): Presents a high level explanation of each of the test scenarios used during the benchmark testing.
- [Performance Analysis](#): Presents an analysis of the results recorded during the benchmark testing.
- [Test Results](#): Presents the actual results of the tests carried out during the benchmark testing.

A number of appendices are included at the end of the document. A brief summary of each is provided below:

- [Portal Configuration](#): Presents a detailed description of the portal environment in terms of the users and groups, site structure and typical content that is presented on each page.
- [Test Environment Specification](#): Presents a detailed description of the test environment used throughout the benchmark testing.
- [System Tuning Guide](#): Presents the detailed configuration settings for the test environment.
- [Deployment Best Practices](#): Presents a number of best practices from the field to assist VAP customers in maximizing the performance of their deployment.
- [Detailed Test Scenarios](#): Presents a step-by-step guide of all the user interactions in each of the test scenarios.

Please refer to these appendices as appropriate and where more detailed information is required in examining the report.

---

---

<sup>2</sup> The use of the word typical does not imply the most common or necessarily recommended production environment but rather serves as an example of a realistic customer deployment of VAP.

---

## Testing Methodology

---

**Portal Deployment** To ensure that the performance results of the benchmark tests are reflective of a realistic customer deployment of VAP, much emphasis has been placed on configuring and deploying the portal in a manner similar to that anticipated in a large-scale enterprise deployment.

To this end, VAP has been installed and configured in a two node clustered environment. The environment consists of a large user base of 1,000,000 users and intricate user group hierarchy consisting of 1,000 groups with a hierarchy of up to 5 levels deep. 50 distinct sites have been created with 26 pages per site. Each page has been enabled for registered user access with 14 of those being additionally enabled for guest user access. Each page consists of approximately 8 modules per page varying in type from standard modules (e.g., Bookmarks, Text Pads, etc.), WebConnector modules exposing content from remote web sites to modules used to manage and present content to users (e.g., Story Publisher and Content Explorer).

More detailed information regarding the deployment of the portal, including screen shots of some of the pages can be found in the [Portal Configuration](#) and [Test Systems Specification](#) appendices at the end of this document.

**Load Testing Tool** All testing has been carried out using the load-testing tool SilkPerformer 5.1 from Segue Software ([www.segue.com](http://www.segue.com)). Please refer to the section on [Test Architecture & Topology](#) for more detailed information as to the set up of SilkPerformer 5.1 during the testing.

**Test Phases** The strategy used during the benchmark testing has been to carry out the performance testing in three phases, each of which is described below:

- Baseline Testing – Baseline testing is used for two primary purposes: First, initial tests ensure that no external factors such as network or server bottlenecks are present, affecting the results of further testing. Second, baseline application testing provides a set of performance results that can be used as a basis for comparing with the actual benchmark testing.
- Tuning – With some overlap between the end of baseline testing and the start of benchmark testing a number of test scenarios are used to determine the optimal configuration of the environment and application, to attain the best results possible from the actual benchmark testing.
- Benchmark Testing – This phase consists of running the pre-defined test scenarios to determine the actual benchmark results. These test scenarios have been carried under a range of operating conditions such as nominal load, increasing workload and peak period usage. Scalability testing on 1, 2 and 4 CPUs has been completed to measure the scalability of VAP and a final longevity test has been carried to determine the stability of VAP over an extended period of time.

The following sections briefly explain the specific methodology used during each phase. More detailed information regarding specific scenarios can be found in the section on [Test Scenarios](#) and the appendix presenting the [Detailed Test Scenarios](#).

**Baseline Testing** All the baseline tests discussed in this document use the “stress test” option in SilkPerformer. A stress test is the execution of a normal test script with no think time between user transactions (i.e., as soon as a response is received by a virtual-user the next request is sent to the server).

This enables the baseline tests to focus on the basic performance of the network, server and VAP framework without any attempt to simulate realistic end user behavior.

**Tuning**

In determining the optimal configuration of the application and environment, a range of parameters were evaluated and tuned to achieve the best performance results. A full list of all the configuration settings in the test environment can be found in the [System Tuning Guide](#) appendix.

It is important to note that additional tuning (not covered by the tuning recommendations in this document) may be required in specific customer environments to attain optimal performance. This section briefly explains some of the more relevant parameters and their effect on application performance:

- Servlet Reload Interval/JSP Page Check Seconds – Both of these parameters determine how often WebLogic will check for newer versions of compiled servlets and JSP pages respectively. The default settings for these are both below 10 seconds, which does not serve any useful purpose in a production environment and can greatly affect the performance of the application. Both of these settings were set to 600 seconds and used throughout the testing.
- Garbage Collection – There are a range of garbage collection parameters that can be used to more efficiently carry out the garbage collection process. Configuring the generational garbage collection parameters increased application performance by approximately 5 – 10% during the tuning process.
- WebLogic Execute Threads – The ExecuteThreads parameter in WebLogic determines the number of threads WebLogic will use to service HTTP requests. Numerous options were tested for this parameter, ranging from 15 – 120 threads. Higher response times and greater CPU utilization was generally observed when using higher values (> 60) for the number of threads. Little difference was experienced when testing in the range of 15 – 60 threads. Consequently it was decided to set the number of threads to 30 for the duration of the testing. It should be noted however; that this setting is very application specific so it is recommended that tuning of this parameter occurs independently in all VAP production environments.

More specific details on these and other parameters can be found in the [System Tuning Guide](#) appendix.

---

**Benchmark Testing**

In contrast to the baseline tests all benchmark testing has been carried out using a mean value of 30 seconds for think time in SilkPerformer. Think time has been used in all benchmark scenarios and is placed between sequential requests in every scenario. This enables the benchmark tests to focus on the performance of VAP in the most realistic usage conditions.

The benchmark testing is also separated into a number of distinct phases to fully analyze the overall performance of VAP:

- Nominal Load – The objective of the nominal load performance testing is to analyze the performance of the portal under non-peak operating conditions.
- Increasing workload – The increasing workload tests are subsequently carried out for a subset of the scenarios with two objectives in mind. Firstly these tests are used to determine the maximum number of concurrent users VAP can support with acceptable response times. Secondly, the tests are used to determine the stability of VAP under extreme load.
- Peak Load – The results of the increasing workload tests are then used to run a series of peak load<sup>3</sup> tests for a subset of scenarios to analyze the performance of VAP during periods of peak usage. The end user load for the peak load tests is ramped up over an initial period of 10 minutes. The results of these tests do not include this ramp up time. The performance metrics are recorded for 60 minutes after this ramp up period.

---

<sup>3</sup> Marginally below peak CPU utilization based on the results of the increasing workload tests.

- Scalability Testing – At peak period usage, a number of tests are carried out to determine the scalability of VAP when comparing the performance of the portal with 1, 2 and 4 CPUs successively.
- Longevity Testing – At a sustained load of approximately 60 – 80% of peak period usage, a longevity test is carried out to determine the stability and consistency of performance of VAP over a period of 8 hours.

All benchmark testing has been carried out with the exact same configuration settings across all tests (i.e., VAP clustered environment). A load balancer is used to balance HTTP requests between the two application servers as part of the scalability testing. To maintain consistency between tests and factor into account any additional latency that may be imposed by the load balancer in the test results, the load balancer has been used for all test scenarios, even those where only 1 application server is being tested. In these scenarios the load balancer has been configured to direct all traffic to the appropriate application server.

## Performance Metrics & Terminology

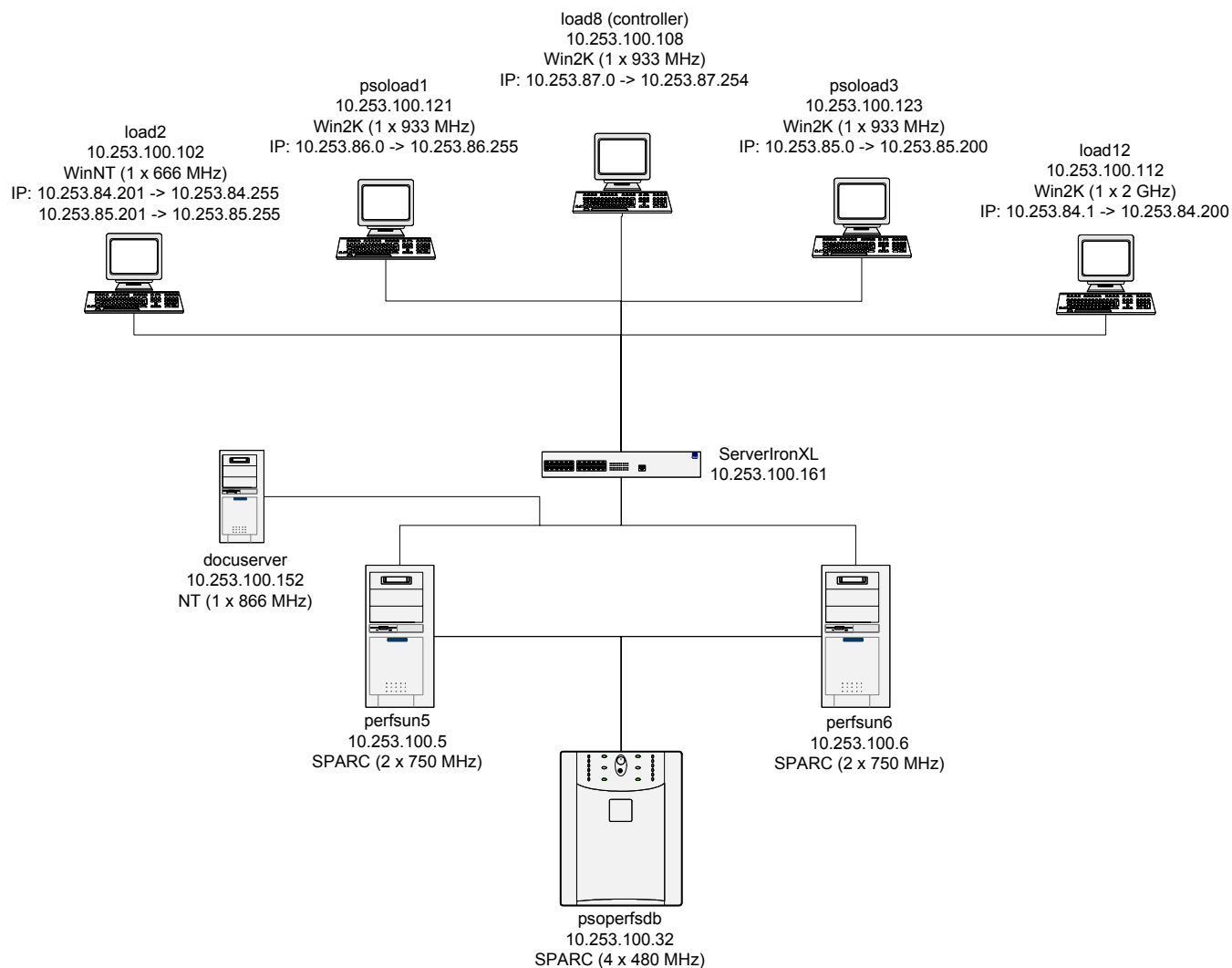
<b>Virtual Users</b>	The number of SilkPerformer virtual users that are used in the test to simulate real user activity. One virtual user may represent many real users depending on the test scenario and the real user behavior.
<b>Think Time</b>	Think time is the time that a virtual user waits before submitting a request for subsequent pages in a test scenario. Think time is typically inserted between each request and is randomly generated, given a mean value for the distribution. Think time is used in an attempt to simulate a more realistic browsing behavior similar to that which a real life end-user may exhibit.
<b>Average Page Load Time (sec)</b>	The total time to load a portal server page with all its elements (including images) in seconds. This measurement represents the performance from the user perspective.
<b>Page Views per second (PV/sec)</b>	Average number of page views processed by the server every second. This can be considered the throughput of the server from a VAP perspective and is the number that best represents the portal server performance.
<b>Transactions per second (Trans/sec)</b>	For some scenarios the metric, transactions per second is used in place of page views per second. A transaction generally consists of more than one page view so the results in transactions per second will include all the pages visited during that transaction.
<b>Transferred Data (KB/sec)</b>	Average amount of data exchanged with the server every second, including header and body content information as well as TCP/IP related traffic. This metric includes both request and response data and represents network throughput.
<b>Http Hits per second</b>	Average number of HTTP requests that are processed by the Web server every second.
<b>CPU Utilization (%)</b>	The total percentage of time that the CPU was busy (includes user, system and all other non-idle time). If there are multiple CPUs per server, this is the average CPU utilization based on individual CPU measurements.

---

## Test Architecture & Topology

---

<b>Introduction</b>	<p>This section discusses the configuration of SilkPerformer and the load balancer and presents a network architecture diagram of the load testing environment including controller, load agents and all servers involved in the test setup.</p>
<b>SilkPerformer</b>	<p>A number of options have been configured in SilkPerformer prior to the beginning of testing. Each of these and its purpose is described in this section.</p> <p>Using the "Automatically load images" option in SilkPerformer all images are downloaded whenever requested in order to test the network aspect of performance as well. However, once an image is downloaded by a virtual user as part of a request for a VAP page, this image is cached for the duration of the transaction the virtual user is executing. This is to simulate the caching mechanism in use by all major browsers.</p> <p>The "First time user" SilkPerformer option is used to generate a realistic simulation of users that visit a web site for the first time. Persistent connections are closed, the Web browser emulation is reset, and the document cache, the document history, the cookie database, the authentication databases, and the SSL context cache are cleared after each transaction. In this case, SilkPerformer always downloads the complete sites from the server, including all files.</p> <p>The VAP schema is populated with 1,000,000 unique test users. Before each virtual user executes a test scenario one of these test users is randomly selected and is used by that virtual user for the duration of the transaction.</p>
<b>External Web Server</b>	<p>An external Web Server is used to provide additional content without having to retrieve content from the internet. The server is used to populate standard WebConnector and some other modules. Performance metrics for the external Web Server are not recorded since the content is cached by these modules once requested and the server is only periodically required to re-serve the content when the cache timeouts have expired.</p>
<b>Network</b>	<p>All servers and load generating agents used in the tests are connected to the same 100 Mbps network segment.</p>
<b>Load Balancer</b>	<p>A load balancer is used for all testing to distribute requests to either 1 or 2 application servers used in the testing. The load balancer has been configured for "sticky" sessions. Consequently, when testing 2 application servers the distribution of traffic to either server is done in a round-robin fashion. Subsequent requests are assigned to the same server based on the IP address of the load agent.</p> <p>Given that 5 load agents (including the controller) are used to generate the load for all testing it was necessary to multi-home each load agent with multiple IP addresses so that traffic would be evenly balanced between both application servers. SilkPerformer supports this by assigning each virtual user a unique IP address from those available on each server to be used in every request to VAP. The network diagram illustrates the range of IP addresses assigned to each individual load agent and the controller.</p>

**Figure 1 – Network Topology Diagram**

## Test Scenarios

### Introduction

A number of test scenarios have been used in both the baseline testing and the benchmark testing. An overview of each of these scenarios and their objectives are described in this section.

The benchmark scenarios give a breakdown of the types of interactions a virtual user will randomly select over the course of the test duration. Where not obvious, these interactions will be briefly described in this section. More detailed information regarding these interactions and an in depth presentation of each scenario, including the specific VAP pages requested can be found in the [Detailed Test Scenarios](#) appendix.

### Baseline Scenarios

#### Static HTML

**Overview:** This scenario continually requests a static HTML page served by WebLogic. The size of the HTML page is 24,197 bytes for the text and 60,652 bytes for all the images.

**Objective:** Measure the throughput of the network to ensure that no network bottlenecks are present, limiting the performance of the application. The second objective of this test is to provide a baseline measurement of the performance of WebLogic serving a static HTML page, representative of the size of a typical VAP portal page being tested as part of the benchmark.

#### Static JSP

**Overview:** This scenario continually requests a static JSP page served by WebLogic. The static JSP page is simply the same static HTML page in the "static HTML" scenario renamed to be a JSP page (i.e., .jsp extension). This ensures that WebLogic parses and serves the page as a JSP page.

**Objective:** Provide a baseline measurement of the performance of WebLogic serving a static JSP page. This baseline measurement represents the best performance achievable by WebLogic under the test conditions and should be used for comparison with the actual benchmark test results.

#### Empty Guest VAP Page

**Overview:** This scenario continually requests an empty VAP page. An empty VAP page is one, which contains no content in the form of modules. Rendering of all other elements provided by the VAP framework are included such as the grid, the header and the footer. The size of the empty VAP page is 41,555 bytes for the text and 34,292 bytes for the images.

**Objective:** Provide a baseline measurement of the performance achievable by the VAP framework. As content is added to the page the performance will degrade, as more resources will be required to render the content on the page and deal with any authorization that may be required. The results of this test should be used in comparison to the overall performance of VAP and some of the earlier baseline tests.

#### Login (Empty VAP Page)

**Overview:** This scenario continually selects a random user and logs that user into a site with an empty VAP home page. At the end of the scenario the user is logged out. An empty VAP page is one, which contains no content in the form of modules. Rendering of all other elements provided by the VAP framework are included such as the grid, the header and the footer. The size of the empty VAP page is 41,555 bytes for the text and 34,292 bytes for the images.

**Objective:** Provide a baseline measurement of the performance of the login process in the VAP framework. The results of this test can be used to compare the login performance of VAP deployments utilizing some custom user or group management.

---

**Benchmark  
Scenarios****Guest User**

**Overview:** This scenario simulates an Internet deployment of VAP. All virtual users simulate guest user access to the portal. Each virtual user executing the transaction will visit a randomly selected site as a guest and carry out a sequence of 10 interactions. Each interaction that is carried out is selected based on the following probabilities:

Page Navigation: 85%  
CAM Links & View File: 15%

**Objective:** Determine the performance of VAP for purely guest user access. Guest users to a VAP site do not have the ability to personalize the site, so this scenario primarily simulates the browsing of content.

---

**Registered  
User**

**Overview:** This scenario simulates an Intranet deployment of VAP. All virtual users simulate registered user access to the portal. Each virtual user executing the scenario will initially visit the guest home page on a randomly selected site. A test user is randomly selected and logged into the appropriate site and a sequence of 10 interactions is carried out before the user is finally logged out. Each interaction that is carried out is selected based on the following probabilities:

Page Navigation: 75%  
CAM Links & View File: 15%  
Move Module: 4%  
Add/Remove Module: 4%  
Add User Prefs: 1%  
Update User Prefs: 1%

**Objective:** Determine the performance of VAP for purely registered user access. In addition to the general viewing of content by registered users, additional interactions have also been included for users to customize the layout and presentation of modules displayed on VAP pages.

---

**Combined**

**Overview:** The combined scenario simulates an Extranet deployment of VAP. All virtual users simulate a combination of guest and registered user access to the portal. At the beginning of each transaction, each virtual user will determine the user type to simulate based on the following probabilities:

Guest: 20%  
Registered: 80%

Selecting either the guest or registered user will result in the virtual user executing the guest and registered user scenarios as described above.

**Objective:** Determine the performance of VAP for a combination of both guest and registered user access. This scenario provides the key benchmark information as it most closely represents a realistic usage scenario for a majority of production applications.

---

---

## Performance Analysis

---

<b>Introduction</b>	This section discusses the performance results from the tests presented in the <a href="#">Baseline</a> and <a href="#">Benchmark</a> Test Results sections.
<b>Baseline Tests</b>	<p>The baseline network test illustrates that ample network bandwidth exists for the performance testing and does not pose a bottleneck for the benchmark testing. Network throughput of up to 6.8 MB/sec was recorded and average response time was below 0.5 seconds for an 84 KB HTML page.</p> <p>The single server static HTML scenario produced 40.7 PV/sec with an average response time of 0.49 seconds. This serves to highlight the upper threshold that is achievable by the application server serving a typical HTML page. The similar test for a static JSP page produced 35.1 PV/sec with an average response time of 0.56 seconds. As expected a slight drop in throughput is observed from the results for the static HTML page due to the additional overhead required by WebLogic in serving the JSP page.</p> <p>Examining the baseline results for the VAP framework revealed 30.3 PV/sec with and average response time 0.65 seconds. This is not a significant drop in throughput or increase in response time from the static JSP page, considering the amount of functionality included in rendering an empty VAP page.</p> <p>The performance of the <a href="#">login</a> (empty VAP page) produced 8.4 Transactions/sec with an average response time of 2.37 seconds per transaction. Application server and database server CPU utilization was recorded at 98.4% and 7.6% respectively. Transactions per second are used in place of page views per second in this scenario to provide a more accurate reflection of actual login performance where a user will typically visit the guest home page, then click on the login link before finally logging into the home page. The transaction described here also includes the process of the user logging out. Also included in the results are the individual page response times for each of the pages in the transaction for a more detailed comparison by interested parties.</p>
<b>Nominal Load Test</b>	<p>When analyzing the results of the <a href="#">nominal load testing</a>, the metric of most relevance is the average response time for pages. The throughput, measured in PV/sec, does not provide much insight into performance at this point due to the fact that it has not peaked and as such will increase as additional load is applied.</p> <p>A nominal load test was carried out for all three benchmark scenarios: Guest User, Registered User and Combined. The average response times at nominal load for all three scenarios were very similar at 0.41 seconds, 0.43 seconds and 0.42 seconds for the guest user, registered user and combined user scenarios respectively.</p> <p>It is important to note that the response times measured in these tests do not include browser-rendering time. The actual page load time may also include different network latencies experienced by disparate users over a corporate LAN or over the internet, that were not present in the configuration used for these tests. As a result of this, perceived response times may vary depending on the end user system configuration and on the network topology between the end user connection and the web / application server.</p>
<b>Increasing Load Test</b>	The increasing load tests are carried out to determine the maximum number of concurrent users that each scenario can support with acceptable average response times. Acceptable average response times are considered to be less than 4 seconds at periods of peak loads.

The actual results presented in the [increasing load test](#) section are the results of the test averaged over the duration of the entire test. This includes the ramp up period and the duration of time that each test runs after the limit of 500 virtual users has been reached. It is for this reason that the average response times are high for these tests. Although the average response times are high after a load of approximately 350 concurrent users has been reached, the tests illustrate that VAP can serve requests and recover once this level of load has been relieved.

The pertinent information from these tests is extracted from the accompanying graphs. These graphs present the application server CPU utilization, the number of virtual users and the average response time for each scenario. The maximum number of concurrent virtual users that can be supported for each scenario is identified by approximately determining the number of virtual users currently executing at the point at which the response times increase above unacceptable levels. This provides a good approximation as to the peak number of concurrent users that each scenario can support within acceptable response times.

Also, of interest from these graphs is the fact that the response times do not go above unacceptable levels until the CPU is nearly fully utilized. This indicates that there are no bottlenecks in the application limiting the performance. The application is in fact CPU bound, which enables the scalability of the overall application.

The graphs reveal the following values for the approximate number of concurrent users that each scenario can support:

Guest User Scenario:	305
Registered User Scenario:	270
Combined Scenario:	285

As each application server is a dual CPU server (750 MHz), these numbers can be easily translated into concurrent users per CPU for capacity planning purposes by dividing each by 2. Clearly, the 750 MHz speed of each processor must be considered when using these figures for capacity planning purposes.

---

### **Peak Usage Load Test**

The maximum number of concurrent users identified from the increasing workload tests, is used to measure the performance of VAP during periods of peak usage for each scenario. The [peak usage load test](#) section presents the results of these tests. The emphasis of these tests is on the performance characteristics and behavior of the system under extremely high load.

For the guest user scenario (305 concurrent users), throughput of 9.3 PV/sec with an average response time of 3.43 seconds was recorded. Application and database server CPU utilization of 94.1% and 14.1% was observed respectively. Performance was consistent for the duration of the test and no errors were encountered.

For the registered user scenario (270 concurrent users), the performance is again consistent for the duration of the test. For registered users the throughput is slightly lower at 8.4 PV/sec with a marginal increase in the average response time to 3.47 seconds. Application and database server CPU utilization of 95.1% and 13.0% were recorded respectively.

For the combined user scenario (285 concurrent users), throughput of 8.6 PV/sec with an average response time of 4.06 seconds was recorded. Application and database server CPU utilization similar to the previous scenarios was recorded at 94.7% and 13.5% respectively.

From a capacity planning perspective, the throughput of 8.6 PV/sec for the combined scenario, can be used in conjunction with the maximum number of concurrent users of 285.

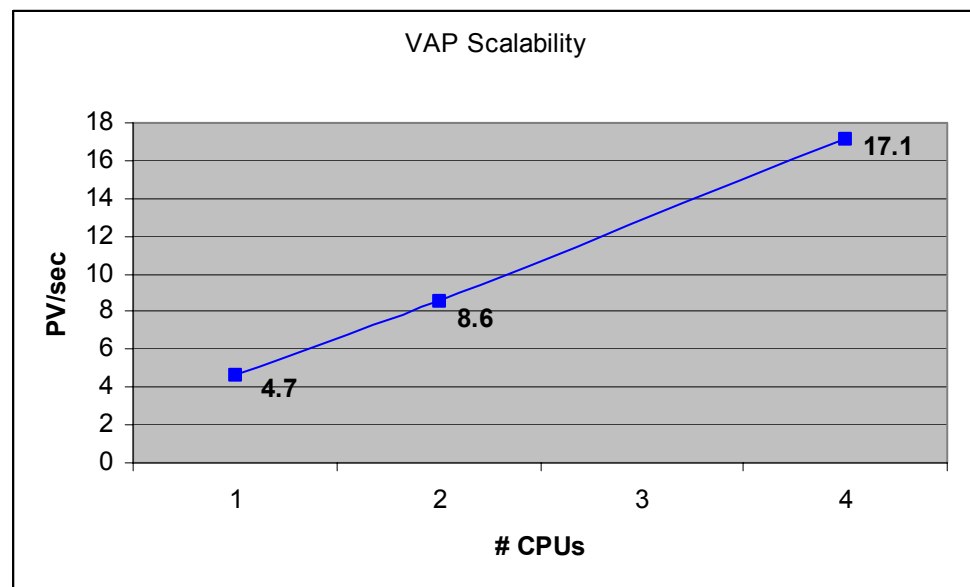
## Scalability

The previously presented peak usage tests have illustrated that VAP is a CPU bound application. This indicates that there are no bottlenecks present, limiting the overall performance of the application, other than the processing power of the CPU itself.

With this in mind, the [scalability testing](#) serves to illustrate that as additional processors are added to the production environment, in an attempt to increase the capacity of the portal, the throughput and number of concurrent users that can be subsequently supported will increase proportionally. An application, which exhibits a near linear proportional relationship between performance improvements, measured by throughput (PV/sec and/or the maximum number of concurrent users) and the number of processors added can be considered to be highly scalable.

Thus, the scalability testing has been carried out using the combined scenario and a series of tests against a 1, 2 and 4 CPU environment. All CPUs were of the same specification i.e. 750 MHz SPARC processors. The number of virtual users to be used in each test was based on the figure of 285 concurrent users for 2 CPUs from the increasing workload test for the combined scenario. The following figure presents the results of the scalability testing for 1, 2 and 4 CPUs.

**Figure 2 – VAP Scalability**



As illustrated in the above figure, VAP supports near linear scalability with an increase in performance from 4.7 PV/sec (143 concurrent users) to 8.6 PV/sec (285 concurrent users) to 17.1 PV/sec (570 concurrent users) for the 1, 2 and 4 CPU test respectively.

It should be noted that the 4 CPU scalability test was carried out using horizontal scalability (2 servers with 2 CPUs each) rather than adding additional CPUs to a single server. It is anticipated that vertical scalability (adding multiple processors to the same server) will exhibit similar scalability to that presented above for up to 4 CPUs. Other considerations must be taken into account when testing a single JVM running on more than 4 CPUs such as how the operating system is effectively able to manage the resource scheduling. Where available, horizontal scalability is the preferred recommended option in terms of scalability, performance, fail over and stability.

Please refer to the "[Best Practices](#)" appendix at the end of this document and the *Vignette Application Portal Scalability and Capacity Planning Overview* that is available on Vignette Global Marketplace (<http://global.vignette.com>) for more information on the advantages / disadvantages of vertical versus horizontal scalability.

---

**Longevity Test**

The [longevity test](#) measures the stability of VAP and the consistency of performance results over a period of 8 hours at approximately 70% of peak usage for the combined user scenario. The test results show consistent performance of 6.5 PV/sec with an average response time of 0.76 seconds over the duration of the test period. Application server and database server CPU utilization of 66.1% and 10.2% respectively is as expected when targeting a usage level of 70% of peak usage.

---

**Capacity Planning**

For capacity planning purposes the benchmark testing has provided two key metrics in estimating individual architectural requirements. These two metrics are the maximum number of concurrent users and throughput, measured in PV/sec. The results can be summarized as follows:

	PV/sec/CPU	# Concurrent Users/CPU
Guest User:	4.6	153
Registered User:	4.3	136
Combined User:	4.3	143

It should be noted that these capacity planning metrics were recorded on 750 MHz Sun SPARC processors and must be adjusted accordingly for the speed of different processors and the architecture of the selected hardware platform.

It should also be noted that these results for the number of concurrent users assume an average think time of 30 seconds for all scenarios. If the assumed think time of 30 seconds is not applicable for certain situations then the above figures for the number of concurrent users (the PV/sec/CPU is not impacted significantly by variations in the think time) can be adjusted accordingly to account for a different think time.

A crude approximation that can be used to calculate the number of concurrent users assuming an average think of Y seconds in place of 30 seconds is as follows:

$$\# \text{ Con. Users (Y secs Think Time)} = (153 / 30) * Y \text{ secs}$$

This calculation provides an approximation. Further performance testing should be carried out to determine more accurate results.

---

## Baseline Test Results

### Baseline Guest Test

<b>Testing Goal</b>	To ensure that no network, server or application bottlenecks are present limiting the overall performance of the application. The results of the baseline tests serve as a comparative threshold on the performance of the application server and VAP application.					
<b>Configuration</b>	The initial Static HTML test is carried out against both application servers (perfsun5 and perfsun6) to measure the network throughout. The load balancer is used to distribute requests between both application servers. All subsequent tests are carried out against the application server perfsun6, to measure the baseline results for a single application server.					
<b>Test Setup</b>	Load Generation Tool	Segue SilkPerformer 5.1				
	Load Agents	5				
	Virtual Users	20 (40 for network test)				
	Test Duration	20 minutes				
	Workload	Steady-state				
	Think Time	0 seconds				
<b>Test Results</b>		<a href="#">PV/sec</a>	<a href="#">Resp. Time (sec)</a>	<a href="#">KB/sec</a>	<a href="#">HTTP Hits/sec</a>	<a href="#">CPU %</a>
	Static HTML (Network – perfsun5 / perfsun6)	79.3	0.49	6856	1587	99.2/ 99.2
	Static HTML	40.7	0.49	3517	814	99.2
	Static JSP	35.1	0.56	3039	702	99.0
	Empty Guest VAP Page	30.3	0.65	1880	455	98.5

## Baseline Login Test

Testing Goal	To measure the performance of the standard login, when logging into an empty VAP page. These results should be used as a basis for comparison when analyzing the login performance of a standard VAP deployment or one, which utilizes custom user or group management.						
Configuration	All tests are carried out against the application server perfsun6, with the database hosted on psoperfsdb.						
Notes	In place of PV/sec, the login test uses transactions/sec to measure the performance of the login process. This is used as it provides a metric of logins per second that a server can support which might be useful. During a transaction a user will typically visit the guest home page, then click on the login link before finally logging into the home page. The transaction described here also includes the process of the user logging out.						
Test Setup	Load Generation Tool	Segue SilkPerformer 5.1					
	Load Agents	5					
	Virtual Users	20					
	Test Duration	20 minutes					
	Workload	Steady-state					
	Think Time	0 seconds					
Test Results		<a href="#">Trans/ sec</a>	<a href="#">Resp. Time (s)</a>	<a href="#">KB/sec</a>	<a href="#">HTTP Hits/sec</a>	<a href="#">APP CPU %</a>	<a href="#">DB CPU %</a>
Individual Pages	Login Transaction	8.4	2.37	1137	260	98.4	7.6
	Guest Home Page		0.66				
	Sign In Page		0.24				
	Process Sign In		1.18				
	Logout		0.28				

## Benchmark Test Results

### Nominal Load Test

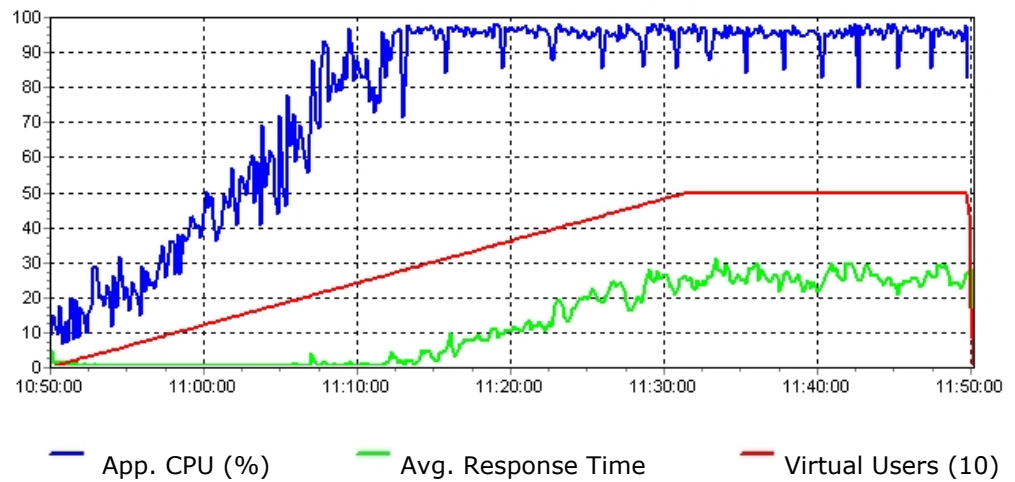
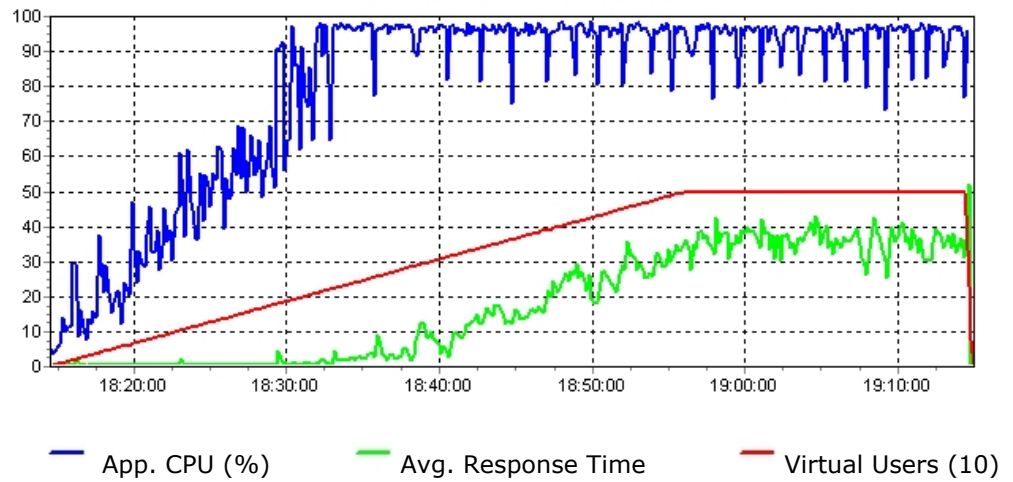
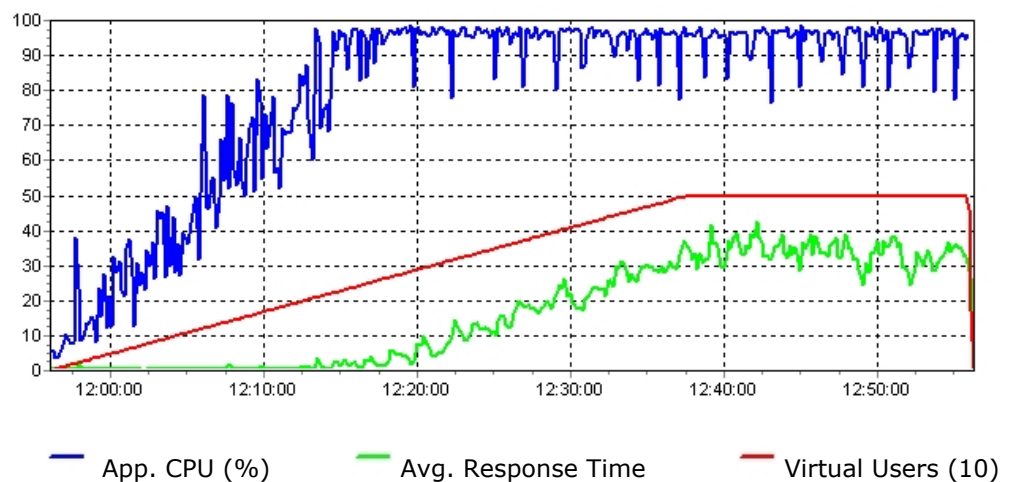
<b>Testing Goal</b>	To measure the performance of the VAP deployment under light load (the server CPU utilization should not be more than 30%). This test emphasizes the user perspective of performance. The primary metric for this test is average page load time.			
<b>Configuration</b>	All tests have been carried out against perfsun6 with traffic being directed through the load balancer. The database is hosted on the server psoperfsdb.			
<b>Test Setup</b>	Load Generation Tool	Segue SilkPerformer 5.1		
	Load Agents	5		
	Virtual Users	100		
	Test Duration	20 minutes		
	Workload	Steady-state		
	Think Time	30 seconds		
<b>Test Results</b>	Test Scenario	Guest	Registered	Combined
	<a href="#">Average Response Time (sec)</a>	0.41	0.43	0.42
	<a href="#">Page Views per second</a>	1.7	1.8	1.7
	<a href="#">Transferred Data (KB/sec)</a>	118	114	108
	<a href="#">Http Hits per second</a>	8	8	8
	<a href="#">App CPU utilization (%)</a>	15.5	17.1	16.1
	<a href="#">DB CPU utilization (%)</a>	4.2	3.6	4.4

## Increasing Load Test

<b>Testing Goal</b>	To determine the maximum number of concurrent users VAP can support for each of the primary scenarios being tested and ensure the stability of VAP under extreme load for a short duration of time (30 – 40 minutes). The graphs presented below are used to determine an approximate value for the maximum number of concurrent users each scenario can support.			
<b>Configuration</b>	All tests have been carried out against perfsun6 with traffic being directed through the load balancer. The database is hosted on the server psoperfsdb.			
<b>Test Setup</b>	Load Generation Tool	Segue SilkPerformer 5.1		
	Load Agents	5		
	Virtual Users	1 – 500		
	Test Duration	60 minutes		
	Workload	Increasing <sup>4</sup>		
	Think Time	30 seconds		
<b>Test Results</b>	Test Scenario	Guest	Registered	Combined
	<a href="#">Average Response Time (sec)</a> <sup>5</sup>	15.07	19.95	19.16
	<a href="#">Page Views per second</a>	7.5	6.8	6.9
	<a href="#">Transferred Data (KB/sec)</a>	548	441	459
	<a href="#">Http Hits per second</a>	40	34	35
	<a href="#">App CPU utilization (%)</a>	77.3	79.6	79.3
	<a href="#">DB CPU utilization (%)</a>	11.4	11.1	11.3

<sup>4</sup> Test is started with 1 virtual user with 2 virtual users added every 10 seconds until a total of 500 virtual users has been reached.

<sup>5</sup> The average response times for the increasing load tests have been averaged across the entire test, from the ramp up period to the extreme load at 500 virtual users. It is for this reason that the response times appear high from these test results.

**Guest User Scenario****Registered User Scenario****Combined User Scenario**

## Peak Usage Load Test

<b>Testing Goal</b>	To measure the performance of VAP at peak usage. Each test is at a steady-state workload with the number of virtual users for each scenario selected based on the results of the increasing workload tests presented previously.			
<b>Configuration</b>	All tests have been carried out against perfsun6 with traffic being directed through the load balancer. The database is hosted on the server psoperfsdb.			
<b>Test Setup</b>	Load Generation Tool	Segue SilkPerformer 5.1		
	Load Agents	5		
	Test Duration	60 minutes		
	Workload	Steady-state <sup>6</sup>		
	Think Time	30 seconds		
<b>Test Results</b>	Test Scenario	Guest	Registered	Combined
	<a href="#"># Virtual Users</a> <sup>7</sup>	305	270	285
	<a href="#">Average Response Time (sec)</a>	3.43	3.47	4.06
	<a href="#">Page Views per second</a>	9.3	8.4	8.6
	<a href="#">Transferred Data (KB/sec)</a>	654	528	549
	<a href="#">Http Hits per second</a>	46	39	39
	<a href="#">App CPU utilization (%)</a>	94.1	95.1	94.7
	<a href="#">DB CPU utilization (%)</a>	14.1	13.0	13.5

<sup>6</sup> The results of these tests have been collected for the steady-state portion of each test (60 minutes). In arriving at this steady state users were initially ramped up for a period of 10 minutes, starting at 1 user and adding 10 users every 10 seconds until the target number of virtual users has been reached. This initial ramp up period is not included in the results.

<sup>7</sup> The number of virtual users selected for each scenario is based on the results of the increasing workload test.

## Scalability Test

<b>Testing Goal</b>	To measure the scalability of VAP. The scalability of VAP is determined by analyzing the results of the combined scenario when executed against 1, 2 and 4 CPUs sequentially. Each test is at a steady-state workload with the number of virtual users for each test selected based on the results of the increasing workload tests adjusted for the number of CPUs being tested against.			
<b>Configuration</b>	1 and 2 CPU tests have been carried out against perfsun6. 4 CPU tests have been carried out against both perfsun5 and perfsun6. All traffic is directed through the load balancer. The database is hosted on the server psoperfsdb.			
<b>Test Setup</b>	Load Generation Tool	Segue SilkPerformer 5.1		
	Load Agents	5		
	Test Duration	60 minutes		
	Workload	Steady-state		
	Think Time	30 seconds		
<b>Test Results</b>	CPU	1 CPU	2 CPU	4 CPU
	<a href="#"># Virtual Users</a> <sup>8</sup>	143	285	570
	<a href="#">Average Response Time (sec)</a>	1.48	3.78	3.9
	<a href="#">Page Views per second</a>	4.7	8.6	17.1
	<a href="#">Transferred Data (KB/sec)</a>	298	543	1060
	<a href="#">Http Hits per second</a>	21	39	77
	<a href="#">App CPU utilization (%)</a> <sup>9</sup>	87.8	93.9	94.3
	<a href="#">DB CPU utilization (%)</a>	7.78	13.4	28.7

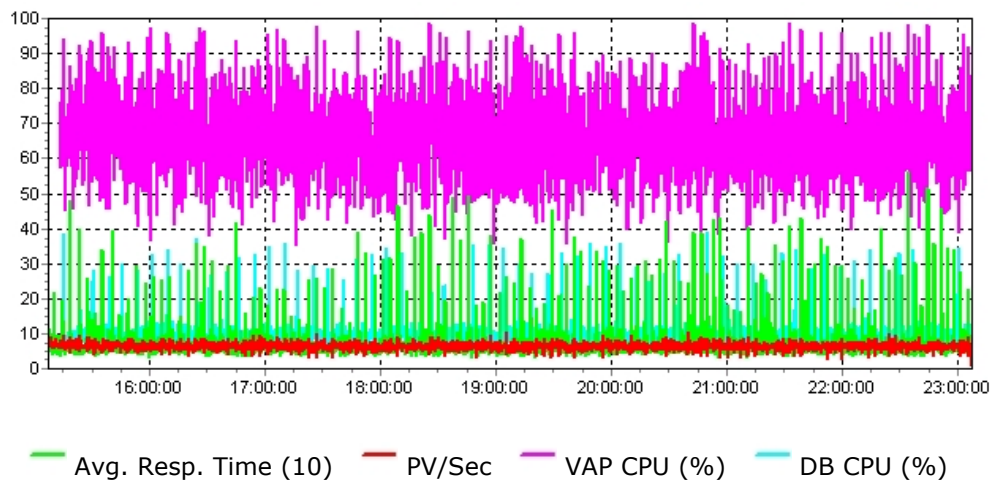
<sup>8</sup> The number of virtual users selected for each scenario is based on the results of the increasing workload test.

<sup>9</sup> The CPU utilization for the application server has been averaged across all CPUs used in each test. For example the value of 94.3% for the 4 CPU test is the average across all 4 CPUs.

## Longevity Test

<b>Testing Goal</b>	To measure the performance of VAP under load over an extended time period. This test helps to determine the stability and consistency of the performance of the system over a period of time.	
<b>Configuration</b>	All tests have been carried out against perfsun6 with traffic being directed through the load balancer. The database is hosted on the server psoperfsdb.	
<b>Test Setup</b>	Load Generation Tool Load Agents Virtual Users Test Duration Workload Think Time	Segue SilkPerformer 5.1 5 200 480 minutes Steady-state 30 seconds
<b>Test Results</b>	Combined Scenario	
	<a href="#">Average Response Time (sec)</a> <a href="#">Page Views per second</a> <a href="#">Transferred Data (KB/sec)</a> <a href="#">Http Hits per second</a>	0.76 6.5 407 29
	<a href="#">App CPU utilization (%)</a> <a href="#">DB CPU utilization (%)</a>	66.1 10.2

### Combined Scenario



# Appendices

## A1 – Portal Configuration

<b>Introduction</b>	This appendix presents detailed information regarding the setup and deployment of Vignette Application Portal (VAP) 4.1 SP1 used throughout the benchmark testing. Screen shots of typical VAP pages used in the benchmark testing are also presented.
<b>VAP Version</b>	4.1 SP1 (build 26)
<b>Licenses</b>	Full "Production Permanent" licenses are used for the clustered installation of VAP
<b>Patches</b>	<p>The following list of patches have been installed in the test environment:</p> <ul style="list-style-type: none"> <li>- <b>Bookmark Module</b> (Bookmark_1_11.car): This patch addresses a problem with the bookmark module when more than one bookmark is added to a module instance as a "required" link.</li> <li>- <b>Content Viewer</b> (ContentViewer_1_10.car): This patch addresses the caching mechanism used by the content viewer module. The caching has been modified to cache content on a "per module" basis rather than on a "per user, per module" basis.</li> <li>- <b>Story Publisher</b> (StoryPublisher_1_11.car): This patch addresses the caching mechanism used in the story publisher module. The caching has been modified to utilize the cluster aware VAP Cache API.</li> <li>- <b>Index on Documents table:</b> A SQL query on the Documents table was incorrectly using a legacy ID column, rather than the new unique ID scheme used in this release of VAP. To correct the performance problem an index on the legacy ID column was created. This problem has been rectified in VAP 4.1 SP2 by modifying the query to use the correct ID column.</li> </ul> <p>The Bookmark module patch and the index on the Documents table have been included in the VAP 4.1 SP2 / SP3 release. If using either the Content Viewer or Story Publisher modules with VAP 4.1 SP2 / SP3, it is recommended opening a support ticket to download both of these.</p> <p>All four of these patches are included with the VAP 4.5 release.</p>
<b>Special Config.</b>	<p>The only special VAP configuration required to execute the test scenarios without error is that the Guest user view is enabled on the Content Explorer module.</p> <p>This is not set by default and can be modified by adding the following line to the PBD file (\$VAP_HOME/config/beans/defs/CM_ContentExplorer.pbd):</p> <pre> &lt;VIEW   ID="explorer"   USERLEVEL="guest"   IS_NAVIGATION_ROOT="true"   PREFERRED_PAGE_ID="generic" &gt; &lt;/VIEW&gt; </pre>

**VAP Setup**

The following table presents exact information as to the overall setup and of the portal:

Category	Allocation
Users per system	1,000,000
Sites per system	50
Groups per system	1000
Group levels	5
Users per group	~500
Groups per user	5
Pages per site	26 pages for registered users 14 guest enabled
Modules per system	1000
Pages per system	1300 = (50x26)
Modules per page	~8
CAM Documents	200,000
CAM Folder levels	10
Navigation Items	26 per site

**Module Setup**

This section presents the module types used and their distribution across VAP pages for the benchmark testing.

In total there are 1,000 module instances distributed across all pages and all sites. Page 1 and Page 2 share some common modules with all other modules distributed over the remaining pages. The breakdown of module instances and their distribution across pages is as follows:

**Page 1:** Bookmark (1), TextPad (1), Story Publisher (1), WebConnector (1), Content Explorer (1), Notes (50), Content Viewer (1).

**Page 2:** Content Directory (1), Content Search (1), Custom News (1), Custom Search (1), Discussion Boards (1), Document Directory (1), News Tracker (1), Content Viewer (1).

**All Other Pages:** Bookmark (100), Text Pad (100), Story Publisher (100), WebConnector (300), JDBC Data Source (1), File Data Source (1), Content Viewer (26), Search the Web (1), Clip (1), Comment Clip (1), Maps (50), Package Tracker (50), Calculator (50), Suggestion Box (50), Survey (1), Task List (50), Web Directory (50), WSC Weather (1), XML Data Source (1).

**Module Configuration**

This section presents the configuration of those modules requiring configuration to achieve the performance results described in this document. No specific module instance configuration was carried out. Configuration, where necessary was carried out similarly across all modules instances of the same type (e.g., all WebConnector modules have identical configuration).

Where no configuration is present in this section for a specific module it can be assumed that the module uses its default configuration. Additionally if a specific parameter is not described in this section, then it is assumed that the value for that parameter is left at the default setting.

**WebConnector:** The timeout on main view and secondary view is set to 15 seconds. All content is set to cache forever. Dirty reads are allowed and persistence is set to on. Cache settings are applied to both main view and embedded binary content.

**Bookmark:** Bookmark modules are configured with 3 unique bookmarks. The ALLOWDUPLICATES custom property is set to true.

**TextPad:** Each module is configured to contain 330 bytes of text.

**Story Publisher:** Story publisher modules have been configured with the Default rule to allow all users to see the provided content. All content is set to expire after 30 minutes.

**Content Explorer:** The content explorer module is set to point to the root (/) folder in the content hierarchy.

**Content Viewer:** Content viewer modules are configured to point to a 1KB text file. Modules are configured to point to an identical file in a different sub folder in the content hierarchy. The cache is set to expire in 30 days.

**Content Directory:** The content directory module is configured to provide links to 3 different sub-folders in the content hierarchy.

**Custom News:** This module is configured to display the default number of 3 headlines from "Accounting News Front Page" and "Advertising" topics.

**Search the Web:** Configured to point to "Hotbot".

**Clip:** Configured to timeout out after 15 seconds and never cache content.

**Comment Clip:** Configured to not timeout on long requests and not to cache content.

## Sample Pages

This section presents some screen shots of selected sample pages from the scenarios used in the benchmark testing.

**Figure 3 – Registered User Home Page**

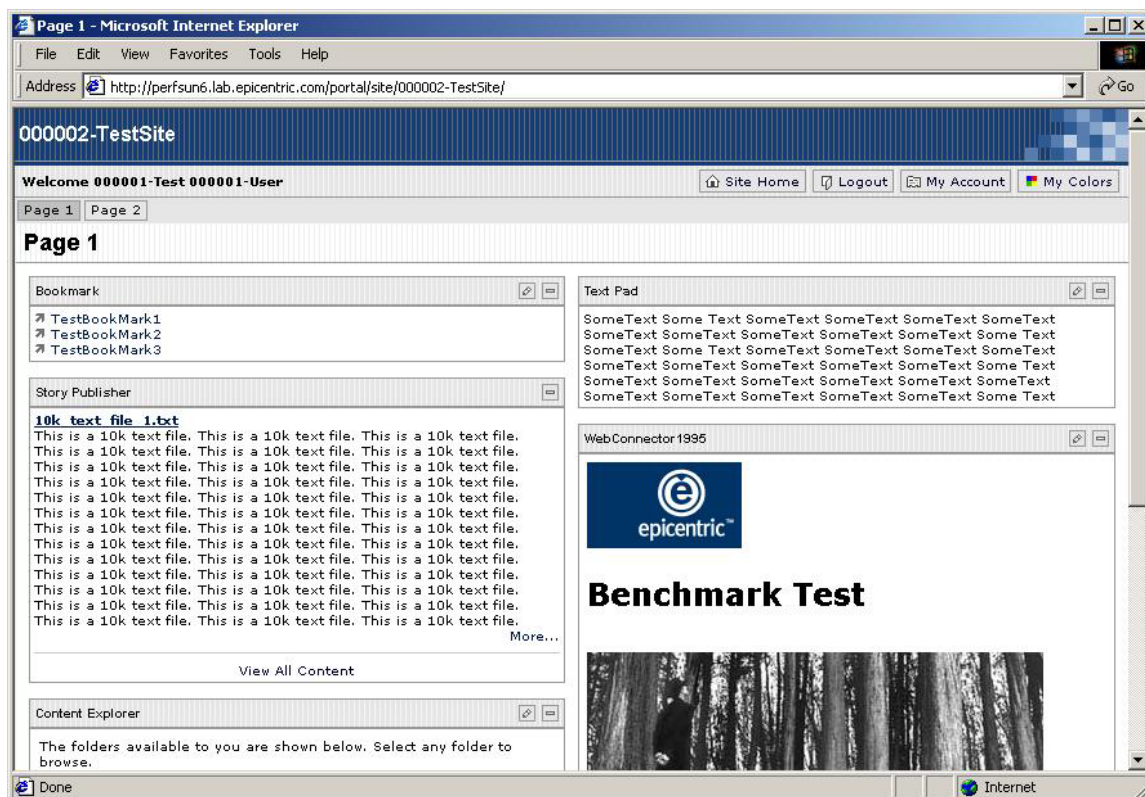


Figure 4 – CAM Content Explorer with Navigation Items

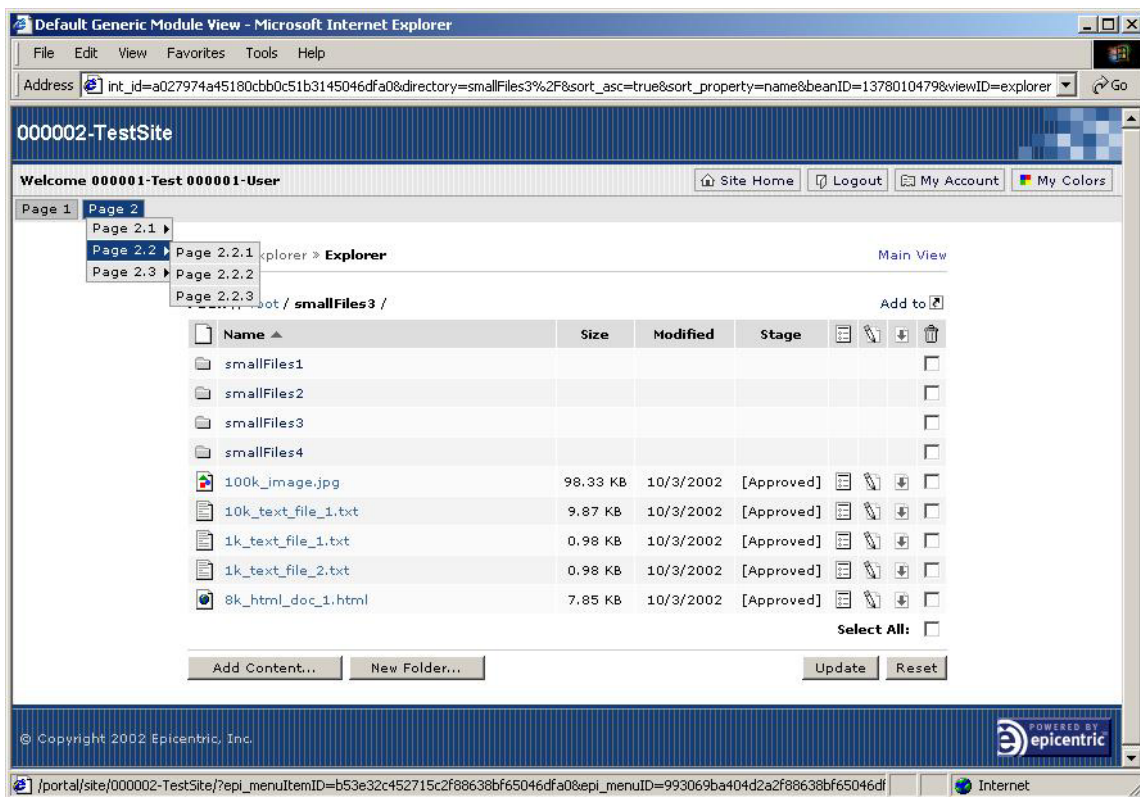


Figure 5 – Guest User Page

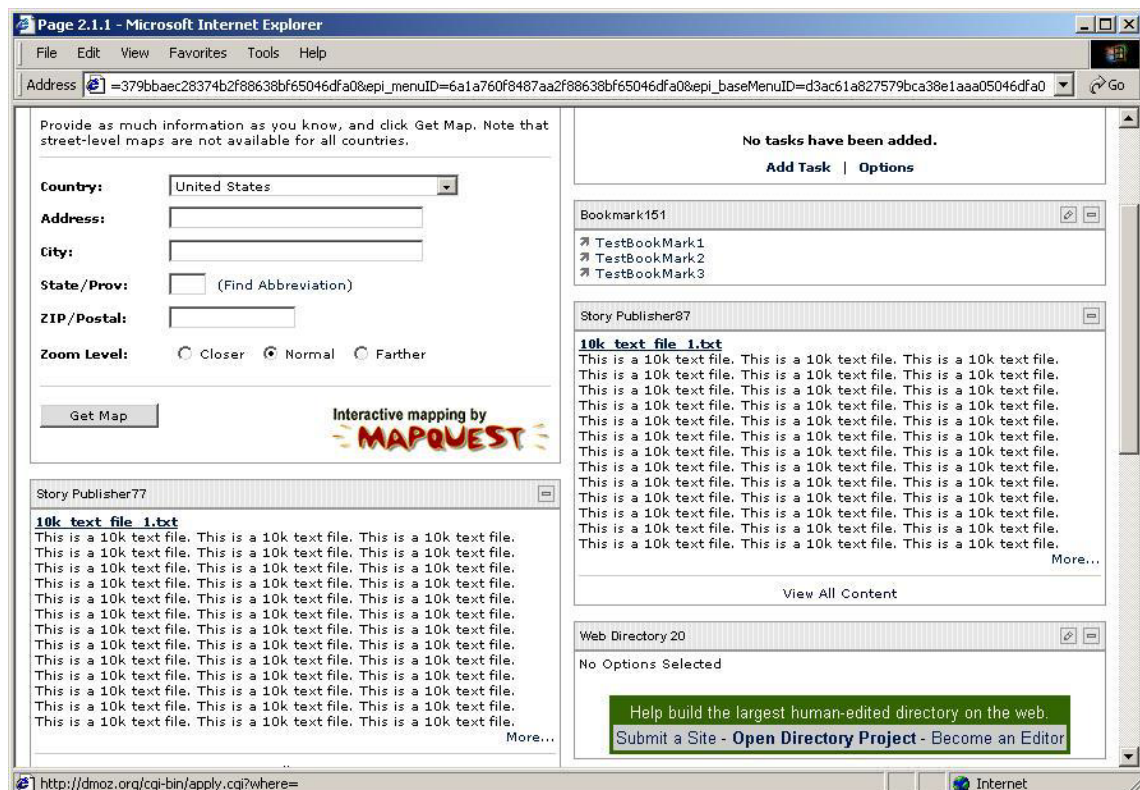


Figure 6 – Add / Remove Modules

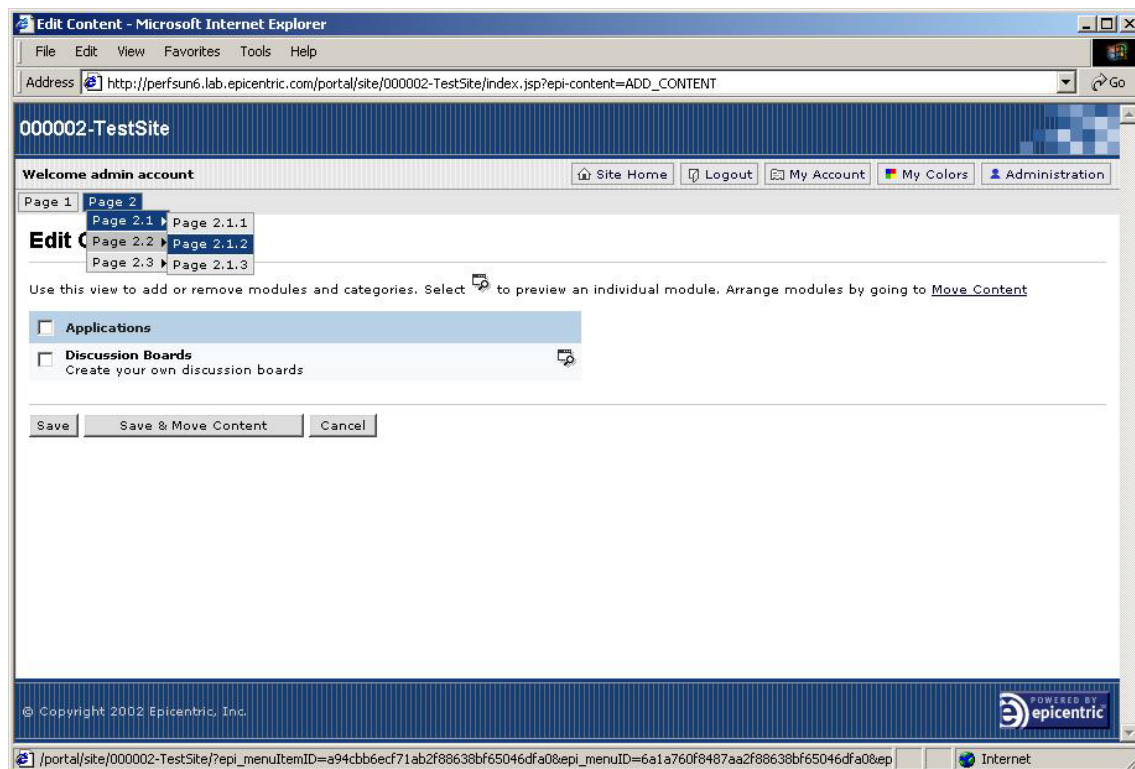
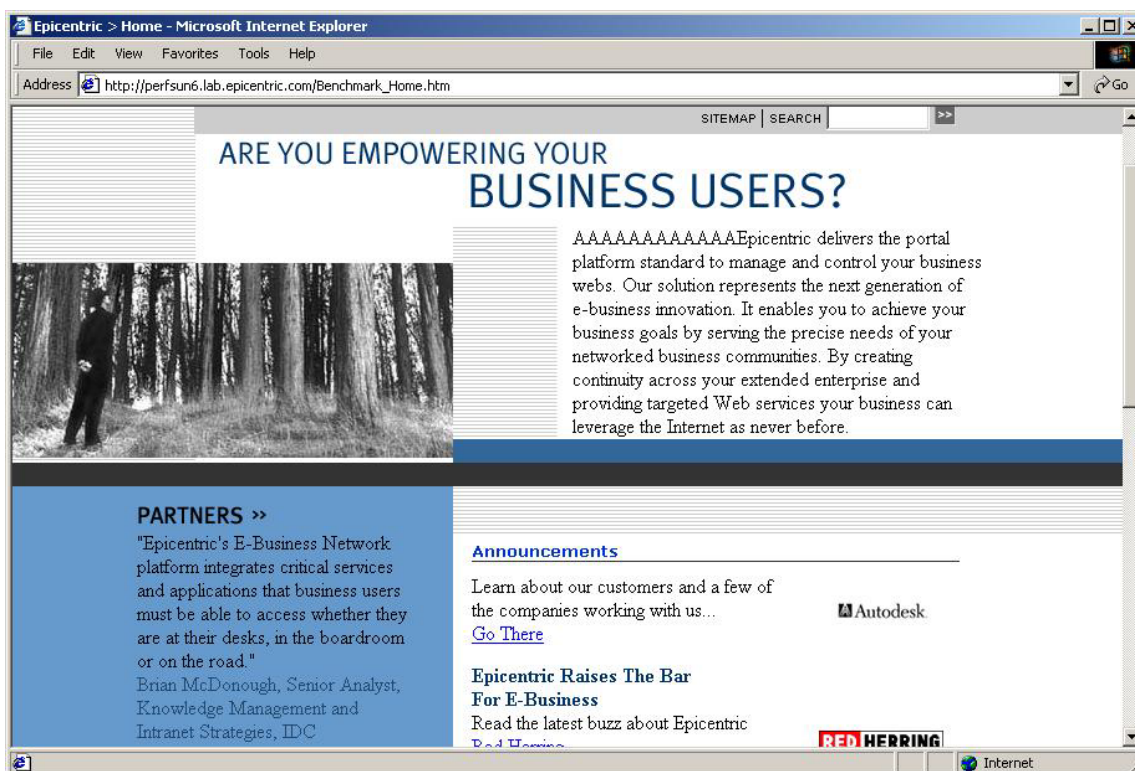
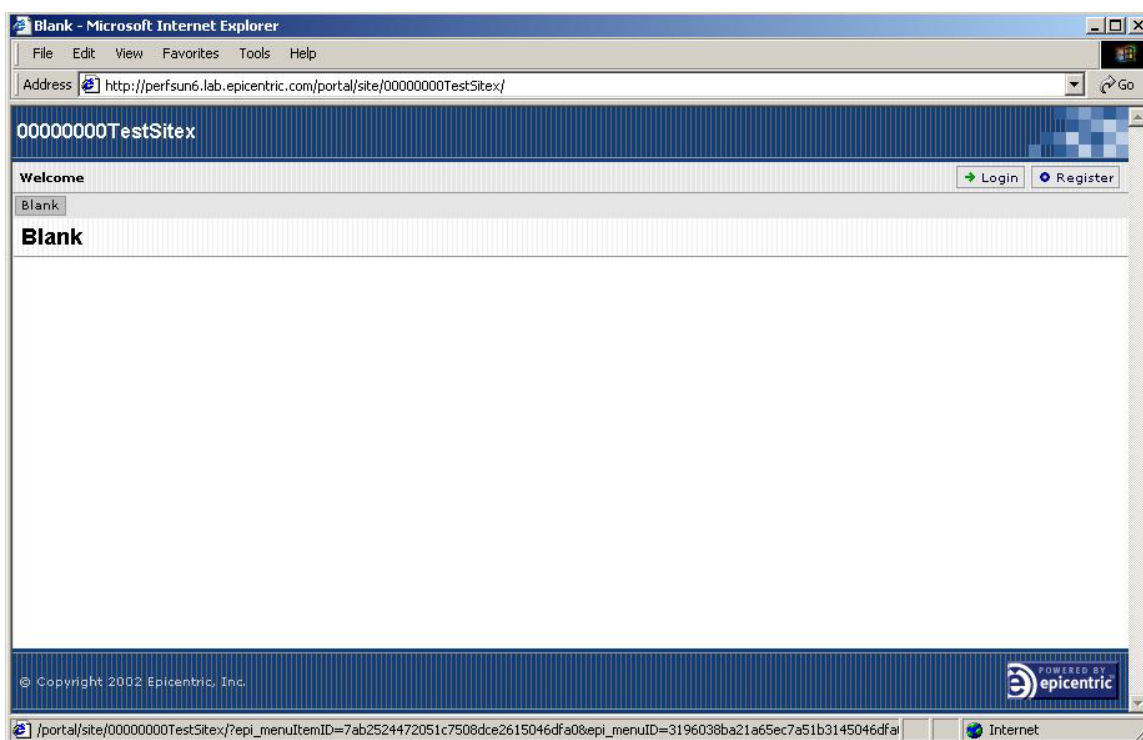


Figure 7 – Static HTML Page



**Figure 8 – Blank VAP Page**

## A2 – Test Systems Specification

**Introduction** This appendix presents the specifications of all the servers used in the benchmark tests including the load balancer, the application servers and external web server, the database server and the load test controller and load agents.

The distribution of virtual users among the load controller and load agents was determined automatically by Silk Performer based on the specification of each load agent i.e. higher specification agents had more virtual users assigned to them.

**Load Balancer** Foundry Networks ServerIronXL (IP Address: 10.253.100.161).

The load balancer is configured for “sticky” sessions based on the IP address of the client. New requests are distributed to a server in a “round-robin” fashion.

### Application Servers

Server Name	perfsun5	perfsun6
Server IP	10.253.100.5	10.253.100.6
Server Model	Sun Fire 280R	Sun Fire 280R
CPU	2 * 750 MHz (SparcV9)	2 * 750 MHz (SparcV9)
Memory	2 GB	2 GB
Operating System	Solaris 8	Solaris 8
Java VM	Sun 1.3.1_03-b03 HotSpot Client VM	Sun 1.3.1_03-b03 HotSpot Client VM
Web Server	WebLogic 6.1 SP3	WebLogic 6.1 SP3
Servlet Engine	WebLogic 6.1 SP3	WebLogic 6.1 SP3
Portal Server	VAP 4.1 SP1 (build 26)	VAP 4.1 SP1 (build 26)

### Database Server & External Web Server

Server Name	psoperfsdb	docuserver
Server IP	10.253.100.32	10.253.100.152
Server Model	Sun Enterprise 450	Dell OptiPlex GX200
CPU	4 * 480 MHz (SparcV9)	1 x 866 MHz (Intel)
Memory	4 GB	256 MB
Operating System	Solaris 8	NT 4.0 SP6
Web Server	-	Apache 1.3
Database	Oracle 8.1.7	-

**Load Agents & Controller**

<b>Name</b>	<b>load8 (ctrler)</b>	<b>psoload1</b>	<b>psoload3</b>	<b>load12</b>	<b>load2</b>
<b>IP</b>	10.253.100.108	10.253.100.121	10.253.100.123	10.253.100.112	10.253.100.102
<b>Model</b>	Dell OptiPlex GX150	Dell OptiPlex GX150	Dell OptiPlex GX150	Dell OptiPlex GX240	Dell OptiPlex GX200
<b>CPU</b>	1 x 933 MHz	1 x 933 MHz	1 x 933 MHz	1 x 2 GHz	1 x 666 MHz
<b>Memory</b>	512 MB	512 MB	512 MB	1 GB	256 MB
<b>OS</b>	Windows 2000	Windows 2000	Windows 2000	Windows 2000	Win NT 4.0 SP6

## A3 – System Tuning Guide

### Introduction

This appendix presents the relevant configuration settings for the production environment and VAP deployment used in the benchmark testing.

Configuring a particular production environment in line with these guidelines does not guarantee the performance results achieved in this document. Some additional configuration, not covered in this guide, may be specifically required for a particular environment to achieve similar performance.

Also, some parameters such as the WebLogic thread count should be independently tuned for a particular environment.

Parameter	Value	Comments
<b>Java Heap Size</b>		Java -Xmx and -Xms arguments are used to set the maximum and minimum (starting) heap size for the JVM.
<b>Per Node</b>	512m – 512m	<p>For both nodes in the cluster the initial and maximum heap size is set to 512 MB. This helps to avoid the cost of allocating additional heap space during the lifecycle of the application.</p> <p>With a large initial heap size the most significant performance hit of increasing the heap size will be incurred during the JVM initialization, avoiding a continuous resizing of the heap size as the application runs.</p> <p>It is very acceptable to use heap sizes of 1 GB (1024m / 1024m) where additional memory resources are required for a specific deployment. No difference in performance was observed when testing either of these settings.</p>
<b>WebLogic Java Garbage Collection</b>	Start 128m Max 128m Ratio 8	<p>The java arguments -XX:NewSize, -XX:NewSizeMax and -XX:SurvivorRatio control the garbage collection within the JVM. Recent changes to the internal Java garbage collection mechanism have resulted in the concept of collections (young and old). Objects with short lives live and die (are collected) quickly in the young collection. The young collection is garbage collected more frequently but with less resource usage than a full garbage collection, thus improving the overall performance of the garbage collection. Full-scale garbage collections still occur but with more time in between.</p> <p>Performance improvements of up to 5 – 10% have been observed in tuning these parameters as part of the benchmark testing:</p> <p>-XX:NewSize=[1/4 heap size] -XX:NewSizeMax=[1/4 heap size] -XX:SurvivorRatio=8</p>

<b>File Descriptors</b>		Having this parameter set too low may cause a "Too many open files" error. This is a critical error and one, which the application will not recover from.
<b>Hard Limit (ulimit -H -n)</b>	4096	The following Solaris kernel parameters should be set in /etc/system file. The server needs to be rebooted after a change. set rlim_fd_cur=4096 set rlim_fd_max=4096
<b>TCP parameters<sup>10</sup></b>		It should be noted that tuning the TCP stack in this manner can be highly dependent on the settings of the other TCP parameters not presented here. All readers should consult with a system administrator to ensure that any changes made at this level do not adversely impact the maintainability or stability of the server.
tcp_conn_req_max_q0	1024	The default maximum number of incomplete (three-way handshake not yet finished) pending TCP connections for a TCP listener. nndd -set /dev/tcp tcp_conn_req_max_q0 1024
tcp_conn_req_min	1	The default minimum value of the maximum number of pending TCP connection requests for a listener waiting to be accepted. nndd -set /dev/tcp tcp_conn_req_min 1
tcp_time_wait_interval	60,000	This parameter tells TCP how long to keep closed connection control blocks. Once the applications complete the TCP connection, the control blocks will be kept for the requested interval (in milliseconds). nndd -set /dev/tcp tcp_time_wait_interval 60000
tcp_fin_wait_2_flush_interval	16,000	This parameter (in milliseconds) limits the time that a connection can stay in the FIN_WAIT_2 state, which is reached if a connection closes actively. nndd -set /dev/tcp tcp_fin_wait_2_flush_interval 16000
tcp_keepalive_interval	90,000	The interval (in milliseconds) specified with this parameter must expire before a keep-alive probe can be sent. nndd -set /dev/tcp tcp_keepalive_interval 90000
<b>WebLogic Thread Count</b>		This value in the config.xml file corresponds to the number of threads that WebLogic assigns to processing HTTP requests.
ExecuteQueue	30	As the server receives requests, they are assigned to a free thread to process the request. When no free threads exists the request is placed on a queue while waiting to be assigned.  <ExecuteQueue Name="default" ThreadCount="30"/>  Readers should tune this parameter for their particular environment as testing has shown this parameter to be highly application dependent.

<sup>10</sup> These parameters are taken from those specified in the "Tuning BEA WebLogic" guide at the BEA website (<http://e-docs.bea.com/wls/docs61/perform/WLSTuning.html#1113098>).

<b>Servlet Reload Interval</b>		This value in the config.xml determines how often WebLogic will check to see if a Servlet has been updated since a previous request.
ServletReload CheckSecs	600	<pre>&lt;WebAppComponent Name="portal" ServletReloadCheckSecs="600"&gt;</pre> <p>This should be set as high as possible in a production environment to provide the best performance.</p>
<b>JSP Page Reload Interval</b>		This value in the weblogic.xml, in the WEB-INF directory determines how often WebLogic will check to see if a JSP has been updated since a previous request. This parameter should be specified as part of the jsp-descriptor tag element.
PageCheck Seconds	600	<pre>&lt;jsp-param&gt;   &lt;param-name&gt;pageCheckSeconds&lt;/param-name&gt;   &lt;param-value&gt;600&lt;/param-value&gt; &lt;/jsp-param&gt;</pre> <p>This should be set as high as possible in a production environment to provide the best performance.</p>
<b>WebLogic Performance Packs</b>	True	<p>The WebLogic performance packs for Solaris should be used. The performance packs use platform-optimized native I/O.</p> <p>This parameter can be modified from the WebLogic console or can be modified directly in the file config.xml:</p> <pre>NativeIOEnabled="true"</pre>
<b>Session Timeout</b>	30	<p>This parameter determines the amount of time that must expire between user interactions before a particular session will expire. The parameter is specified in the web.xml file in the WEB-INF directory.</p> <pre>&lt;session-config&gt;   &lt;session-timeout&gt;30&lt;/session-timeout&gt; &lt;/session-config&gt;</pre>
<b>WebLogic Connection Backlog Buffering</b>	1024	<p>This parameter specifies how many TCP connections can be buffered in a wait queue. This queue is populated with requests for connections that the TCP stack has received but the application has not accepted yet. This is a fixed size queue definable by this parameter.</p> <p>This parameter can be modified from the WebLogic console or can be modified directly in the file config.xml:</p> <pre>&lt;Server AcceptBacklog="1024" ListenPort="80" Name="myserver" ...&gt;</pre>

<b>VAP Metastore Caching</b>	Users folder	<p>VAP provides full and selective in-memory metastore caching for both non-clustered and clustered VAP installations. Full metastore caching is enabled by default. For any clustered, load balanced deployment, the recommended practice is to set only the users folder to be cached. This can be achieved by adding the following to the properties.txt file:</p> <p>metastore.default.cachefolder.0=/users/</p> <p>The way that the user folder cache consistency is maintained in a cluster environment is that when a user transitions from one JVM to another, a broadcast is sent out, telling the other JVMs in the cluster to drop their cache for that user.</p>
<b>VAP Connection Pool</b>		Parameters for default database connection pool that can be set in VAP properties.txt configuration file.
poolsize	50	<p>Maximum number of connections available in the pool (default = 5).</p> <p>connectionpool.default.poolsize=50</p>
lowerthreshold	10	<p>Minimum number of connections in the pool (default = 0).</p> <p>connectionpool.default.lowerthreshold=10</p>
timeout	60	<p>Connection timeout in seconds (default = 60). A process that is waiting for a connection will not receive a timeout exception until this amount of time has expired.</p> <p>connectionpool.default.timeout=60</p>
lease	45	<p>Connection lease time in seconds (default = 45). If a process does not give back a connection during this amount of time, the connection is discarded and a new one is created. This property reduces the performance impact caused by processes that take connections and never give them back. If a process gives back a connection after its lease time has expired, the connection pool simply kills it.</p> <p>connectionpool.default.lease=45</p>
sleep	10	<p>Connection sleep time in minutes (default = 10). The connection pool breaks a connection if it is not used for this amount of time.</p> <p>connectionpool.default.sleep=10</p>
<b>Database Settings</b>		These parameters can be configured in the properties file for the Oracle database instance (\$ORACLE_HOME/dbs/init[SID].ora)
Sessions	125	<p>The number of Oracle sessions (processes) for the VAP deployment has to be large enough to accommodate the maximum number of connections for all connection pools pointing to the VAP database plus some additional sessions for system processes.</p> <p>processes=125</p>
Cursors	2,000	<p>This parameter specifies the maximum number of open cursors (handles to private SQL areas) a session can have at once.</p> <p>open_cursors=2000</p>

---

Block Buffers	65,536	This parameter specifies the number of database buffers in the buffer cache. It is one of several parameters that contribute to the total memory requirements of the SGA of an instance. db_block_buffers=65536
Shared Pool	536,870,912	This parameter specifies (in bytes) the size of the shared pool. The shared pool contains shared cursors, stored procedures, control structures, and other structures. Larger values improve performance in multi-user systems. Smaller values use less memory. Sufficient memory should be available on the server to set this to the appropriate setting. shared_pool_size=536870912
Sort Area Size Retained Size	10,485,760 10,485,760	These parameters specify in bytes the maximum amount of memory Oracle will use for a sort. After the sort is complete, but before the rows are returned, Oracle releases all of the memory allocated for the sort, except the amount specified by the sort_area_retained_size parameter. After the last row is returned, Oracle releases the remainder of the memory. Sufficient memory should be available on the server to set this to the appropriate setting sort_area_size=10485760 sort_area_retained_size=10485760

---

## A4 – Performance Best Practices

<b>Introduction</b>	This appendix introduces some best practices and recommendations for analyzing performance problems in a typical production environment.
<b># Modules Per Page</b>	<p>It is generally recommended that the number of modules per page should be kept below 10.</p> <p>As expected, the performance of a particular VAP page is highly dependent on the specific type and nature of the modules on that page. There is however a marginal overhead involved on the part of the VAP framework for displaying and authorizing a particular user access to each module view on the page. As such, to minimize this overhead and avoid its impact on performance, it is recommended that no more than 10 modules should be displayed on each page, particularly those pages which end users most frequently access.</p>
<b>Scalability</b>	<p>Scalability of an application is particularly important for large enterprise deployments of VAP where an individual server cannot service the anticipated load on the application. In such cases, VAP provides the opportunity to cluster multiple instances of the application. This can be achieved by either installing multiple instances of VAP on a single server with a large number of CPUs and having each instance run in a separate Java Virtual Machine (JVM) or installing single instances of VAP on separate servers or finally, a combination of the proceeding.</p> <p>As the benchmark results show, VAP scales near linearly from 1 to 2 CPUs on the same server and for 4 CPUs across 2 servers.</p> <p>Vertical scalability can be described as the scalability of an application as additional CPUs are added to the same server. Horizontal scalability can be described as the scalability of an application as additional servers are added to the environment.</p> <p>Even though not tested as part of the benchmark report, it is anticipated that VAP scales equally as well with up to 4 CPUs on the same server (vertical scalability). If VAP is to be deployed on a server with more than 4 CPUs, some testing should be done to determine the optimal performance of the application in this situation. For example, on an 8 CPU server, there may be some performance improvements by running two instances of VAP across 8 CPUs rather than 1 instance across 8 CPUs. The reasons for this are more related to how the operating system, application server and JVM handles the scheduling of threads across a larger number of CPUs. This is also subject to change as later versions of JVMs are released to market. In addition, careful tuning of the application server is required to ensure that the load on a single instance can be most effectively serviced e.g. WebLogic thread count parameter and BackLog parameter. Consequently, when scaling VAP vertically (above 4 CPUs), some performance testing should be carried out to determine the optimal deployment of the application.</p> <p>For horizontal scalability, the only limitation on the scalability of VAP will most likely be from the network or database perspective (the database is a shared resource between all instances of VAP in a particular cluster).</p> <p>With a large number of VAP instances (more than 8) running in a clustered environment it should be ensured that the network throughput being achieved is below 70% of the maximum network bandwidth available at peak periods (i.e. maximum of 8 MB/sec on a 100 Mb/sec network). Network throughput levels above this will impact the average response times for end users during these periods. If this is the case it is recommended to segment the VAP instances onto different network segments over a 1 Gb/sec backbone.</p>

Similarly, in a large VAP clustered deployment, it should be ensured that the database resources at peak periods can adequately support the number of instances in the cluster. Database CPU utilization of above 75% at peak periods can greatly impact the performance of VAP depending on the types of queries being executed at that time.

With these caveats in mind, it is recommended that the scalability of VAP be handled horizontally rather than vertically if the number of production CPUs is more than 4. Horizontal scalability is assured to provide the best scalability and also has additional advantages of fail over and redundancy in an enterprise deployment. The only significant drawback to achieving scalability in this way is the cost and maintenance overhead in such a situation.

---

**Database Server**

For large enterprise deployments of VAP, it is recommended that a dedicated database server be used to host the VAP schema. In addition, certain database subsystems (e.g., metastore, users) can be further distributed across several database servers.

This ensures that the database will not pose a bottleneck as VAP is scaled to support a continually growing user base.

---

**Web Server**

If a web server (e.g. Apache, iPlanet, IIS) is used to proxy requests to the application server in a production environment such as that which may be encountered in an Internet deployment with a DMZ (Demilitarized Zone), it is recommended, where possible, to host the web server on a separate server from the application servers.

The net affect of this is that more resources, in the form of memory and CPU, will be available to the application server to service requests. This will incur a negligible impact on the response times due to additional network hop but will increase the overall throughput of the application due to the additional CPU resources that will be available.

---

**Application Server  
Admin Node**

Similarly to the recommendation for the web server, it is recommended, that in the cases where the deployment requires a separate application server administration node, which will not service end user requests, that this be hosted on a separate server.

For example, in a VAP deployment, which leverages the advantages of both a WebLogic cluster and a VAP cluster it is recommended that the WebLogic administration node not be included as part of the cluster servicing end user requests.

Therefore, separating the administration node onto a separate server will free up memory and CPU resources on the original server and result in an overall increase in throughput for the entire deployment.

As the administration node in the case of WebLogic will not be servicing end user requests the server it is hosted on can be of a much smaller processing power.

---

**Image Size(s)**

It is recommended that some effort is placed in ensuring that any images used in the design for the portal i.e. in the styles, grids, themes and custom modules, be as small in size (KB) as possible, while maintaining the required aesthetic appeal.

This will ensure the least impact on available network bandwidth and will result in the most efficient servicing of HTTP requests for images where the application server itself is required to serve the images. Typically, when compared to a web server such as IIS or Apache, an application server will serve images much less efficiently.

---

By ensuring that the images are as small as possible, it will eliminate unnecessary overhead on the application server and will free CPU resources, resulting in an overall increase in throughput at peak periods.

---

#### **Hardware Architecture**

Much debate surrounds the advantages / disadvantages of the Wintel hardware architecture versus the Sun Sparc hardware architecture. No endorsement of either is implied in this document, as VAP has been developed to run efficiently on all platforms where a supported JVM and application server combination is available. Additionally, much more factors such as cost, corporate policies, security, maintenance and support are involved when making such decisions and cannot be reliably considered in this context.

However, this section has been included to present some findings in this area. Generally, a comparison of the Wintel platform versus the Sun Sparc platform on a pure **cost basis** i.e. where servers of each architecture in a similar price range have been used in performance testing, the Wintel platform has shown much better performance.

---

#### **Custom Module Development**

When developing custom modules for integration with VAP it is imperative that these modules perform optimally so as not to degrade the performance of VAP overall.

In every case, the customer should consider the available options provided by the modules with VAP to determine if they can support the user requirements. This will ensure a supported environment where performance problems can be more easily identified and resolved.

In the situations where this is not possible, it is advisable that customers take advantage of as much of the performance enhancing mechanisms provided by the VAP framework. These mechanisms are discussed in this section.

**Caching:** VAP provides an advanced caching API, which offers a simple cluster safe way to cache application data. The API can cache any application data that can be encapsulated in a serializable java object. This could include content retrieved from remote data sources such as legacy systems, proprietary databases or the file system. Where appropriate, caching should be utilized as often as possible and data should be cached for as long a period of time as is permissible.

**Threading:** VAP provides a threading API, which can be used for modules, which take a significant period of time to render their content. From a performance perspective, this is very useful for pages where one or more custom modules on the page affect the overall performance of the page. To avoid the VAP framework unnecessarily waiting for these modules to complete their processing it is possible to utilize the threading API to carry out the processing of the module asynchronously (occurring in parallel) with the rendering of the other modules on the page.

**Connection Pooling:** VAP provides connection pooling for both database and LDAP connections. Using connection pooling can greatly improve the scalability of an application and improve the overall performance at high loads. For this reason, any custom module development that requires access to resources of these types should leverage the functionality provided by the VAP connection pools. New connection pools for custom development can be easily configured in the appropriate configuration files for VAP.

---

**Identifying  
Performance  
Problems**

Where it is suspected that performance problems are present with a deployment of VAP, there are some simple tests that can be carried out to determine where the cause of the performance problems may be. These steps are presented in this section.

In identifying a performance problem it is necessary to determine where the performance problem may lie. In many situations poor performance is caused by several specific problems.

The recommended steps are as follows:

- Eliminate the network: Many times improper network configuration can cause significant performance problems, resulting in poor response times and low CPU utilization. Network problems may exist at any point in the architecture (e.g., between load balancer and application servers or between application servers and database servers). An easy test to determine if the network is problematic is to run a load test against a static HTML page on the web server or application server, to ensure that high network throughput is achieved. Other free tools can be found online, which can be used to measure network throughput between two servers ([www.netiq.com](http://www.netiq.com)).
- Identify a specific server: Assuming the network is not impacting the performance, each individual server in the cluster should be tested to determine if a specific server is responsible for the poor performance. This may be caused by a simple mis-configuration on the server or a more serious problem such as faulty hardware.
- Run a baseline test: Run a baseline test against a blank VAP page to determine if the performance is as expected. This will ensure that the performance of the core VAP framework is as expected. Refer to the baseline results for this page, in this document, as a comparison.
- Identify specific VAP pages: If the baseline test has run as expected and all servers are performing consistently, the specific pages responsible for the poor performance should be identified. Depending on the load testing tool used, this can be either easy to achieve or may require more extensive work. Due to the fact that the baseline test has performed as expected, this indicates there must be specific pages that are responsible for the poor performance.
- Identify specific modules: After identifying specific pages that are causing performance problems, if not obvious, it will be necessary to identify the specific modules on that page that are responsible. This can be achieved by adding modules successively to the blank VAP page used in the baseline test and test each module in isolation. These tests will quickly identify the offending modules. At this point a thorough analysis of the module itself will be required to determine what in particular is causing the performance problems. The work involved in this is outside the scope of this document.

To summarize, when attempting to identify performance problems the best strategy is to simplify the test scenario. This is the goal of the steps outlined above.

---

## A5 – Detailed Test Scenarios

### Introduction

This appendix presents the detailed steps in each of the scenarios described earlier. In total there have been 6 scenarios used during the testing:

- **Static HTML Page**
- **Static JSP Page**
- **Empty Guest VAP Page**
- **Login (Empty VAP Page)**
- **Guest User Scenario**
- **Registered User Scenario**
- **Combined User Scenario**

In the next section an overview and the objective of each scenario will be presented again with a more detailed breakdown of the steps involved in each. The final section in the appendix will present the procedures or interactions invoked from each scenario.

## Scenarios

### Static HTML Page

**Overview:** This scenario continually requests a static HTML page served by WebLogic.

**Objective:** Measure the throughput of the network to ensure that no network bottlenecks are present. Provide a baseline measurement of the performance of WebLogic serving a static HTML page.

**Steps:**

- Repeatedly call the "[HTML Page](#)" procedure.

### Static JSP Page

**Overview:** This scenario continually requests a static JSP page served by WebLogic.

**Objective:** Provide a baseline measurement of the performance of WebLogic serving a static JSP page.

**Steps:**

- Repeatedly call the "[JSP Page](#)" procedure.

### Empty Guest VAP Page

**Overview:** This scenario continually requests an empty guest VAP page. An empty VAP page is one, which contains no content in the form of modules.

**Objective:** Provide a baseline measurement of the performance achievable by the VAP framework.

**Steps:**

- Repeatedly call the "[Empty VAP](#)" procedure.

### Login (Empty VAP Page)

**Overview:** This scenario continually selects a random user to login to a site with an empty VAP home page. An empty VAP page is one, which contains no content in the form of modules.

**Objective:** Provide a baseline measurement of the performance of the login process for comparison to standard VAP deployments and those utilizing some custom user or group management.

**Steps:**

1. Set the site to be "00000000TestSitex"
  2. Randomly select a user to login.
  3. Call the "[Empty VAP](#)" procedure.
  4. Click on the "Login" link.
  5. Submit the username/password for the selected user, which subsequently loads the blank VAP home page for this site.
  6. Click on the "Logout" link.
- 

**Guest User Scenario**

**Overview:** This scenario simulates an Internet VAP deployment with all guest user access to the portal.

**Objective:** Determine the performance of VAP for purely guest user access.

**Steps:**

1. Call the "[Initialization](#)" procedure.
2. Call the "[Guest Page](#)" procedure.
3. Randomly select from the following procedure calls based on the following probabilities:

"[CAM Navigation & File View](#)" – 85%

"[Page Navigation](#)" – 15%

If page navigation is selected then the page to navigate to is randomly selected based on the following probabilities:

Page 1	- 20%
Page 1.1	- 16%
Page 1.1.1	- 16%
Page 2	- 16%
Page 2.1	- 16%
Page 2.1.1	- 16%

4. Repeat step 2, ten times.
- 

**Registered User Scenario**

**Overview:** This scenario simulates an Intranet VAP deployment with all registered user access to the portal.

**Objective:** Determine the performance of VAP for purely registered user access.

**Steps:**

1. Call the "[Initialization](#)" procedure.
2. Call the "[Guest Page](#)" procedure.
3. Call the "[Login](#)" procedure.
4. Randomly select from the following procedure calls based on the following probabilities:

"[CAM Navigation & File View](#)" – 75%

"[Page Navigation](#)" – 15%

"[Move Module](#)" – 4%

"[Add/Remove Module](#)" – 4%

"[Add User Preferences](#)" – 1%

"[Update User Preferences](#)" – 1%

---

If page navigation is selected then the page to navigate to is randomly selected based on the following probabilities:

Page 1	- 20%
Page 1.1	- 16%
Page 1.1.1	- 16%
Page 2	- 16%
Page 2.1	- 16%
Page 2.1.1	- 16%

5. Repeat step 3, ten times.
6. Call the "[Logout](#)" procedure.

### Combined User Scenario

**Overview:** The combined scenario simulates a VAP deployment with a combination of guest and registered user access to the portal.

**Objective:** Determine the performance of VAP for a combination of both guest and registered user access.

**Steps:**

1. Call the "[Initialization](#)" procedure.
2. If user type is "Guest" then follow the guest user scenario as outlined above. If user type is "Registered" then follow the registered user scenario as outlined above. The probability of guest user type versus registered user type as selected in the initialization procedure is as follows:

Guest User Scenario	- 20%
Registered User Scenario	- 85%

## Procedures

### Initialization

#### Description

The initialization procedure is invoked at the start of every transaction, for every virtual user when executing the benchmark scenarios: Guest User, Registered User and Combined User Scenario. The initialization procedure involves the following steps:

1. Randomly select the site to be used in the test from the range 1 – 50. This is used to populate the SITENUM variable in subsequent requests.
2. Randomly select the test user to access the site. The test user is selected based on the registered test users who have access to the site as selected in step 1.
3. Randomly select the user type based on the following probabilities:
 

80% - Registered User
20% - Guest User

If the user type is registered user, then the user name selected in step 2 will be used as part of the login process. If the user type is guest user then the user name selected in step 2 will be ignored.
4. Execute the selected scenario (Guest User, Registered User and Combined User Scenario).

**HTML Page (1 page)**

Page	URL	Action / Info
HTML Page	http://HOST/Benchmark_Home.htm	The size of the HTML page is 84,849 bytes (24,197 bytes text / 60,652 bytes images).

**JSP Page (1 page)**

Page	URL	Action / Info
JSP Page	http://HOST/Benchmark_Home.jsp	The size of the JSP page is 84,849 bytes (24,197 bytes text / 60,652 bytes images).

**Empty VAP Page (1 page)**

Page	URL	Action / Info
Empty VAP Page	http://HOST/portal/site/00000000TestSite/index.jsp	The VAP page does not contain any content in the form of modules. The VAP framework renders the grid, header and footer as part of the request. The size of the page is 75,847 bytes (41,555 bytes text / 34,292 bytes images).

**Guest Home Page (1 page)**

Page	URL	Action / Info
VAP Guest Home Page	http://HOST/portal/site/SITENUM/index.jsp	This procedure sends a request for the guest home page of the site specified by SITENUM.

**Login (2 pages)**

Page	URL	Action / Info
VAP Login Page	http://HOST/portal/site/SITENUM/index.jsp?epi-content=LOGIN	Click on "Login" hyperlink at current page, thus loading the login page. Enter previously generated username/password and submit the login form. This will authenticate the request and redirect to the registered user home page for the site.
VAP Registered User Home Page	http://HOST/portal/site/SITENUM/index.jsp	

**Logout (1 page)**

Page	URL	Action / Info
VAP Logout Page	http://HOST/portal/site/SITENUM/index.jsp?epi-content=LOGOUT	Click on "Logout" hyperlink at current page, thus logging the user out and loading the logout page.

**Page Navigation (1 page)**

Page	URL	Action / Info
VAP Page (1, 1.1, 1.1.1, 2, 2.1 or 2.1.1)	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on the previously selected random page link, thus navigating to the appropriate page.

**CAM Navigation & File View (5 – 9 pages)**

Page	URL	Action / Info
VAP Page 1	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on "Page 1" link from current page.
Content Explorer Folder View	http://HOST/portal/site/ SITENUM /index.jsp?epi-content=GENERIC ...	Click on the "Folder" image link to view the secondary page of the content explorer. Randomly select how many levels of CAM folders to view from the range 1 – 4. Repeat the following page request for this number of times. For each iteration, the page request is generated randomly by concatenating the string "smallFiles" with a random number selected from the range 1 – 4. When the page request is generated load this sub folder by clicking on the generated link.
Repeat: Content Explorer Sub Folder View.	http://HOST/portal/site/ SITENUM /index.jsp?epi-content=GENERIC ...	A subsequent subfolder is loaded for each iteration in the previous loop. The actual subfolder selected is based on the randomly generated link in each iteration. After the iterations into the sub folders are completed, the 10KB file in the folder is selected to be viewed.
File View	http://HOST/portal/site/ SITENUM /index.jsp?epi-content=GENERIC ...	Load the current site home page. The current site has been previously selected at the start of the overall transaction.
VAP Home Page	http://HOST/portal/site/SITENUM/index.jsp	

**Add User Preferences (5 pages)**

Page	URL	Action / Info
VAP Page 1	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on "Page 1" link from current page.
Edit User Bookmark Module	http://HOST/portal/site/ SITENUM /index.jsp?epi-content=GENERIC&viewID=USER_BEAN_EDIT_VIEW ...	Click on the "Edit" link for the bookmark module. Add a new bookmark (www.anewbookmark.com) and click on "Update". This saves the new bookmark for that user in the Metastore and loads the main view of the module.
VAP Page 1	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on the "Edit" link for the bookmark module.
Edit User Bookmark Module	http://HOST/portal/site/ SITENUM /index.jsp?epi-content=GENERIC&viewID=USER_BEAN_EDIT_VIEW ...	Delete the newly created bookmark (www.anewbookmark.com) and click on "Update". This deletes the new bookmark and loads the main view of the module.
VAP Page 1	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	

**Update User Preferences (5 pages)**

Page	URL	Action / Info
VAP Page 1	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on "Page 1" link from current page.
Edit User TextPad Module	http://HOST/portal/site/SITENUM/index.jsp?epi-content=GENERIC&viewID=USER_BEAN_EDIT_VIEW&beanID=842697930	Click on the "Edit" link for the TextPad module. Choose a color for the text and click on "Save". This saves the updated text for that user in the Metastore and loads the main view of the module.
VAP Page 1	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on the "Edit" link for the TextPad module.

Edit User TextPad Module	http://HOST/portal/site/SITENUM/index.jsp?epi-content=GENERIC&viewID=USER_BEAN_EDIT_VIEW&beanID=842697930	Reset the color for the text and click on "Save". This saves the updated text for that user in the Metastore and loads the main view of the module.
VAP Page 1	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	

**Move Module (7 pages)**

Page	URL	Action / Info
VAP Page 2	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on "Page 2" link from current page. Click on the "Move Content" link.
Move Content Page	http://HOST/portal/site/SITENUM/index.jsp?epi-content=MOVE_CONTENT	Move module to right hand side panel.
Move Content Page	http://HOST/portal/site/SITENUM/index.jsp?epi-content=MOVE_CONTENT	Click on "Done", which loads main view of the page where the user came from.
VAP Page 2	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on the "Move Content" link.
Move Content Page	http://HOST/portal/site/SITENUM/index.jsp?epi-content=MOVE_CONTENT	Move module back to left hand side panel.
Move Content Page	http://HOST/portal/site/SITENUM/index.jsp?epi-content=MOVE_CONTENT	Click on "Done", which loads main view of the page where the user came from.
VAP Page 2	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	

**Add / Remove Module (4 pages)**

Page	URL	Action / Info
VAP Page 2.2.2	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	Click on "Page 2.2.2" link from current page. Click on the "Edit Content" link.
Move Content Page	http://HOST/portal/site/SITENUM/index.jsp?epi-content=MOVE_CONTENT	Select the module (Discussion Boards) to be added/removed from the page and click on "Save", which loads main view of the page where the user came from.
VAP Page 2.2.2	http://HOST/portal/site/SITENUM/index.jsp?epi_menuItemID=PAGE_GUID...	